Министерство транспорта и связи Украины
Государственный департамент по вопросам связи и информатизации
Одесская национальная академия связи им. А. С. Попова

Кафедра информационных технологий

ИНФОРМАТИКА

Модуль 4
Программирование задач со списками и файлами. Объектно-ориентированное программирование

Часть 2

Методические указания
к лабораторным и практическим работам

УТВЕРЖДЕНО
методическим советом факультета
информационных систем
Протокол № 5 от 22.11.2007 г.

Одесса 2008

УДК 004.43

План УМИ 2007 г.

Составители: **И.Г. Швайко, Ю.В. Прокоп, Л.Л. Леоненко, Н.В. Северин**

Ответственный редактор – **Ю. В. Прокоп**

Методическое пособие содержит краткие теоретические сведения и примеры составления программных проектов средствами C++ Builder для решения задач со списками и файлами, а также для создания компонентов и классов. Данная (вторая) часть пособия будет полезной для студентов при подготовке к лабораторным и практическим занятиям по дисциплине «Информатика» в модуле 4 и содержит контрольные вопросы и индивидуальные задания разных уровней сложности для выполнения на компьютере.

Предназначено для приобретения навыков программирования студентами разных специальностей академии, изучающих дисциплину «Информатика», с целью дальнейшего использования ими этих навыков в повседневной нынешней и будущей профессиональной деятельности; также будет полезным для пользователей персональных компьютеров, желающих научиться программировать в среде C++ Builder.

ОДОБРЕНО
на заседании кафедры
информационных технологий
и рекомендовано к печати

Протокол № 2 от 01.10.2007 г.

СОДЕРЖАНИЕ

Предисловие

C++ Builder – популярная среда разработки приложений для операционной системы Windows. Среда объединяет средства языка программирования C++ и компонентно-ориентированный подход к разработке программ. Сочетание простоты освоения визуальной среды разработки и поддержка широчайшего спектра технологий делают C++ Builder универсальным инструментом разработки приложений любого уровня сложности.

В практике программирования важную роль играют списки, обеспечивающие гибкую и эффективную работу с однотипными данными, количество которых заранее неизвестно или изменяется в процессе работы программы. Файловые потоки предоставляют возможность программного создания и изменения файлов на магнитных носителях. Объектно-ориентированное программирование позволяет создавать сложные программные системы.

В данном методическом пособии представлены краткие теоретические сведения, примеры составления программных проектов средствами C++ Builder для решения задач со списками и файлами, а также для создания компонентов и классов, контрольные вопросы и варианты индивидуальных заданий. Эта (вторая) часть пособия будет полезной для студентов при подготовке к лабораторным и практическим занятиям по дисциплине «Информатика» в модуле 4.

Каждая из предложенных к выполнению лабораторных работ имеет задания разных уровней сложности. Студент сам или по указанию преподавателя выбирает задания того или иного уровня сложности согласно порядковому номеру фамилии студента в списке группы. В дальнейшем при оценивании знаний преподаватель может учитывать уровень сложности выполненной лабораторной работы, оптимальность алгоритма составленной программы, своевременность подготовки и выполнения работы.

Перед выполнением лабораторного задания студенту необходимо:

- ✓ уточнить у преподавателя индивидуальное задание;
- ✓ выучить соответствующие разделы теоретического курса согласно лекционным записями и учебной литературе;
- ✓ составить схемы алгоритма решения задачи;
- ✓ изобразить форму проекта и составить таблицу значений свойств ее компонентов;
- ✓ написать тексты программ в C++ Builder;
- ✓ подготовить протокол выполнения лабораторной работы и представить его преподавателю для проверки.

К выполнению лабораторной работы допускается студент, имеющий подготовленный самостоятельно заполненный протокол лабораторной работы.

Содержание протокола лабораторной работы:

- ✓ название темы и цель лабораторной работы;
- ✓ краткое изложение основных теоретических положений;
- ✓ ответы на контрольные вопросы;
- ✓ схемы алгоритмов для решения индивидуального задания;
- ✓ форма проекта и таблица значений свойств ее компонентов;
- ✓ тексты программ в C++ Builder;
- ✓ результаты вычислений на компьютере.

Структура модуля 4

Дисциплина “Информатика” изучается в I – II семестрах и предназначена для обучения студентов работе на персональном компьютере с целью его использования в будущей профессиональной деятельности.

Программа курса состоит из четырех модулей:

модуль 1 – Основные сведения о персональном компьютере и организации вычислительных процессов;

модуль 2 – Программирование задач с циклами и массивами;

модуль 3 – Программирование задач со структурированными типами данных;

модуль 4 – Программирование задач со списками и файлами. Объектно-ориентированное программирование.

Целью изучаемого модуля «Программирование задач со списками и файлами. Объектно-ориентированное программирование» является формирование у студентов таких знаний и умений:

Знания по темам:

- ✓ списки;
- ✓ текстовые и бинарные файлы;
- ✓ объектно-ориентированное программирование.

Умения:

Разрабатывать и выполнять на компьютере алгоритмы и программы для решения задач:

- ✓ создание списка, очереди, стека; просмотр их элементов;
- ✓ выполнение различных вычислений с элементами списка;
- ✓ добавление и удаление элементов списка;
- ✓ создание и просмотр текстовых и бинарных файлов;
- ✓ выполнение различных действий со строками в текстовом файле;
- ✓ работа с датой и временем;
- ✓ обработка структур, хранящихся в текстовом файле;
- ✓ обработка структур, хранящихся в бинарном файле;
- ✓ создание собственного компонента на базе имеющихся;
- ✓ создание класса.

В соответствии с учебным планом структура модуля 4 имеет вид:

Вид занятий	Количество часов
Лекции	10
Практические занятия	10
Лабораторные работы	18
Всего аудиторного времени:	38
Индивидуальная и самостоятельная работа студентов	70
Всего часов:	108

1 Динамические структуры данных. Списки, очереди, стеки

1.1 Динамические структуры данных

Динамическая структура данных – это набор однотипных элементов, которые размещаются в динамической памяти. Примером ранее изученной динамической структуры данных является динамический массив. Кроме динамических массивов, широко применяется ещё одна динамическая структура данных **- список**.

1.2 Список

Списки используются вместо массивов для хранения и обработки однотипных данных, количество которых заранее неизвестно и может изменяться в процессе работы.

Элементы списка располагаются в памяти хаотично. Это даёт возможность беспрепятственно добавлять или удалять любое количество элементов. Для того чтобы иметь возможность работать с элементами как со связанной структурой, каждый элемент должен «знать», где расположен следующий за ним элемент.

Таким образом, кроме данных, каждый элемент должен хранить ещё одно (или два) значения: адрес следующего элемента и/или предыдущего элемента. Естественно представить элемент списка в виде структуры, которая содержит одно или несколько информационных полей (полей данных) и указатель на следующий элемент. После последнего элемента других элементов нет – это означает, что указатель на элемент, следующий после последнего, пустой, иначе говоря, равен 0, или NULL.

Список, в котором хранятся целые числа, может быть объявлен так:

```
struct Element{
   int d;              //целое число
   Element *next;  //указатель на следующий элемент
};
```

Указатель на первый элемент этого списка объявляется так:

```
Element *first;
```

Числовое значение первого элемента: first->d, адрес второго элемента: first->next.

Обратите внимание, что first является не элементом списка, а указателем на него, т. е. является указателем на структуру. Поэтому обращение к полям структуры происходит не через точку, а через стрелочку **->** (символы «-» и «>»).

Список для удобства и наглядности можно изобразить следующим образом (рис. 1.1):

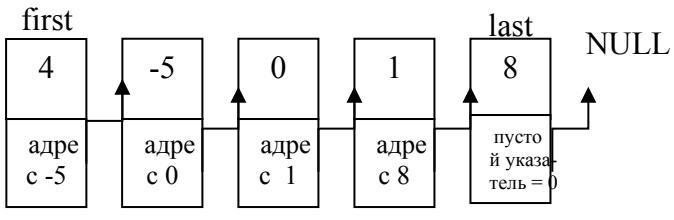


Рис. 1.1

Для прохода по списку используется вспомогательный указатель ***c***. Сначала нужно «встать» на первый элемент:

```
Element *c=first;
```

Затем в цикле, пока список ещё не закончен (т. е. пока указатель на элемент ***c*** не равен 0), с очередным элементом списка выполняется требуемое действие, после чего выполняется переход к следующему элементу списка:

```
while(c!=0){        //пока не конец списка
      ….            //выполняем некоторые действия с c->d;
   c=c->next;       //переходим к следующему элементу списка
}
```

Наиболее простыми разновидностями списков являются стек и очередь.

1.3 Стек

Стек – это список, в который элементы можно добавлять и удалять только в конец. Для стека существует специальный термин – FILO (“First In – Last Out”, т. е. «первым пришёл – уйдёшь последним»).

Объявление стека аналогично объявлению обычного списка:

```
struct Elem{
  int d;            //d – числовое поле
  Elem *next;       //next – указатель на предыдущий элемент
};
```

Для работы со стеком нужно хранить указатель на вершину стека, которая объявляется следующим образом:

```
Elem *root=0;
```

Операции со стеком

Добавление нового элемента над вершиной стека. Последовательность действий:

1 Выделить место в памяти под новый элемент (он пока никак не связан с другими элементами стека):

```
c=new Elem;
```

2 Присвоить полям данных нового элемента требуемые значения (например, новый элемент будет иметь значение 77):

```
c->d=77;
c->next=0;
```

3 Связать первый элемент стека с новым элементом (т. е. полю next нового элемента присвоить указатель на вершину стека):

```
c->next=root;
```

4 Вершиной становится новый элемент:

```
root=c;
```

Удаление верхнего элемента стека. Последовательность действий:

1 Присвоить адрес вершины стека вспомогательному указателю:

```
c=root;
```

2 Указателю на вершину присвоить указатель на второй элемент:

```
root=root->next;
```

3 Освободить память, на которую указывает вспомогательный указатель:

delete c;

Просмотр содержимого стека (например, для вывода на экран). Последовательность действий:

1 Встать на вершину стека (вспомогательному указателю присвоить указатель на вершину):

c=root;

2 В цикле, пока не конец стека (т. е. пока вспомогательный указатель не станет равным 0):

while (c!=0)

a) Вывести значения информационных полей очередного элемента (т.е. элемента, на который указывает вспомогательный указатель):

Form1->Memo2->Lines->Add(IntToStr(c->x));

b) Перейти к следующему элементу. Для этого вспомогательному указателю нужно присвоить адрес, который хранится в его поле next:

c=c->next;

Ниже представлен пример программы с использованием стека.

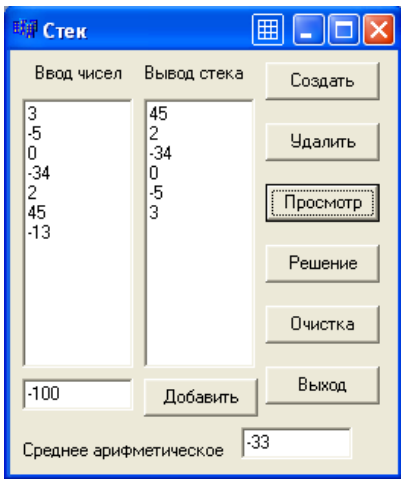


Рис. 1.2

Пример 1.1

Заполнить стек числами из Memo. Пользователь должен иметь возможность добавления и удаления элемента в стек. Вычислить среднее арифметическое чётных значений стека. Удалить элементы стека от вершины до первого элемента с положительным значением.

В данном примере объявление стека и его вершины будет глобальным, т. е. его нужно разместить в самом начале программы до всех подпрограмм.

На рис. 1.2 представлено окно формы после заполнения списка числами из Memo (обратите внимание, что элементы добавляются в начало стека) и добавления нового элемента (нажаты последовательно кнопки «Создать», «Добавить», «Решение» и «Просмотр»).

Текст программы:

```
//Объявление элемента стека – объявление структуры Elem
//Теперь Elem – это тип, который может быть использован наряду
//со стандартными типами
struct Elem{
        int d;          //d – числовое поле
        Elem *next; //next – указатель на предыдущий элемент
};
```

```
//Объявление вершины стека и начальное обнуление
Elem *root=0;

//Обратите внимание, что root – это указатель на структуру, поэтому
//обращение к полям элемента происходит не через «точку», а через
//«стрелочку» (минус больше):
//root->d – числовое значение элемента
//root->next – адрес второго элемента стека (предыдущего элемента)
//---------------------------------------------------------------------------
//Добавление элемента со значением x к стеку
void add (int x){
//Объявляем и размещаем в памяти вспомогательный элемент c
  Elem *c=new Elem;
  c->d=x;            //В числовое поле заносим значение x
  c->next=root;      //Предыдущим перед новым элементом становится
                     //прежняя вершина
  root=c;            //После этого новый элемент становится вершиной
}
//---------------------------------------------------------------------------
//Удаление верхнего элемента стека
void del( ){
  Elem *c=root;      //Запоминаем первый элемент в переменной c
 root=root->next;    //Второй элемент становится первым
                     //(root->next – обращение ко второму элементу)
  delete c;          //Освобождаем память (удаляем первый элемент)
}
//---------------------------------------------------------------------------
//Вывод стека
void print( ){
  Form1->Memo2->Clear( );
  Elem *c=root;                  //Становимся на вершину
  while (c!=0){                  //Пока стек не закончился,
        Form1->Memo2->Lines->Add(IntToStr(c->d));   //Вывадим в Memo
                                 //числовое значение элемента стека
        c=c->next;               //Переходим к нижнему элементу стека
  }
}
//---------------------------------------------------------------------------
//Кнопка или пункт меню «Добавить»
void __fastcall TForm1::Button1Click(TObject *Sender)
{
 add(StrToInt(Edit1->Text));   //Вызываем подпрограмму add
                               //чтобы добавить в стек число из Edit1
}
//---------------------------------------------------------------------------
```

```
//Вычисление среднего арифметического чётных значений стека
float srednee( ){
  int s=0, k=0;
  float sr;
  Elem *c=root;
  while (c!=0){
        if (c->d % 2==0){          //Если значение элемента стека чётное,
              s+=c->d;             // то добавляем его к сумме
              k++;}
        c=c->next;
  }
  if (k!=0)sr=1.*s/k;
  return sr;
}
//---------------------------------------------------------------------------
//Кнопка или пункт меню "Создать"
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  int n=Memo1->Lines->Count;
  for (int i=0; i<n; i++)    //Каждое число по очереди добавляем к стеку
        add(StrToInt(Memo1->Lines->Strings[i]));
}
//---------------------------------------------------------------------------
//Кнопка или пункт меню "Удалить"
void __fastcall TForm1::Button3Click(TObject *Sender)
{
  del( ); //Вызываем подпрограмму удаления вершины
  print( ); //Вызываем подпрограмму вывода стека в Memo2 (для контро-
ля)
}
//---------------------------------------------------------------------------
//Кнопка или пункт меню "Решение"
void __fastcall TForm1::Button5Click(TObject *Sender)
{
  if (root==0){       //Если указатель на вершину равен 0, то стека нет
    //Выводим сообщение, что стек пустой, и прерываем подпрограмму
        ShowMessage("Стек пустой");
        return;}
  Edit2->Text=FloatToStr(srednee( ));
  //Пока вершина не равна 0 (т. е. стек ещё не пустой) и значение вершины
  //неположительное,
  while(root!=0 && root->d<=0)
        del( );          //удаляем вершину
}
//---------------------------------------------------------------------------
```

```
//Кнопка или пункт меню "Просмотр"
void __fastcall TForm1::Button4Click(TObject *Sender)
{
  print( );
}
//---------------------------------------------------------------------------
//Кнопка или пункт меню "Очистка"
void __fastcall TForm1::Button6Click(TObject *Sender)
{
  //Очистка памяти
  while (root!=0) del( );
  //Очистка формы
  Memo1->Clear( );
  Memo2->Clear( );
  Edit1->Clear( );
  Edit2->Clear( );
}
```

1.4 Очередь

Очередь – это разновидность списка, имеющая следующую особенность: новые элементы можно добавлять только в конец очереди, а удалять можно только первый элемент очереди. Для очереди существует специальный термин FIFO (“first in – first out”, т. е. «первым пришёл – первым ушёл»).

Для очереди определены следующие операции:
добавление элемента в конец очереди;
удаление первого элемента очереди;
просмотр содержимого очереди.

В примере 1.2 рассмотрены подпрограммы перечисленных выше операций.

Пример 1.2

Создать очередь из названий городов и их населения (в тысячах человек). Предоставить пользователю возможность добавления и удаления элементов. Определить город с наибольшим населением. Удалить все города до города с наибольшим населением.

Предположим, что в очередь добавлено четыре города и их население: Одесса, Киев, Днепропетровск, Харьков (обратите внимание, что города добавляются в конец очереди). Окно формы после добавления этих городов и нажатия кнопок «Решение» и «Просмотр» представлено на рис. 1.3: определён го-

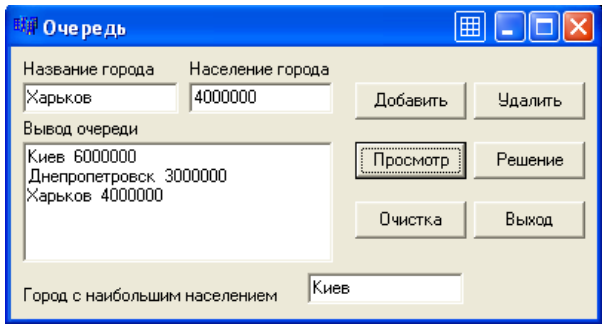


Рис. 1.3

род с наибольшим населением (Киев) и удалены все города до него (из начала), т. е. Одесса.

Текст программы:

```
//Объявление элемента очереди
struct Elem{
        AnsiString gorod; //Строка – название города
        int nas;          //Целое число – население города
        Elem *next;       //Указатель на следующий элемент очереди (адрес)
};
//Объявление указателей на начало и конец очереди и их начальное обнуление
Elem *first=0, *last=0;
//---------------------------------------------------------------------------
//Добавление элемента в очередь
void add (AnsiString S, int d){
//Создание указателя на элемент очереди и выделение памяти
 Elem *c=new Elem;
 c->gorod=S;       //Заносим в новый элемент название города, которое
                   //хранится в переменной S
 c->nas=d;         //Заносим в новый элемент население
 c->next=0;        //После нового элемента пока других элементов нет
 if (first==0)     //Если очереди ещё нет,
        first=c;   //то новый элемент становится первым,
 else              //иначе добавляем элемент после последнего,
        last->next=c;// записывая в last адрес нового элемента
  last=c;          //Теперь новый элемент должен стать последним
}
//---------------------------------------------------------------------------
//Удаление первого элемента очереди
void del( ){
  Elem *c=first;   //Становимся на первый элемент
  first=first->next; //В первый записываем тот, который ранее был вторым
  delete c;        //Уничтожаем удаляемый элемент
}
//---------------------------------------------------------------------------
//Вывод очереди
void print( ){
  Form1->Memo1->Clear( );
//Становимся на первый элемент очереди
  Elem *c=first;
  while (c!=0){    //Пока не конец очереди,
                   //выводим в Memo1 название города и его население
        Form1->Memo1->Lines->Add(c->gorod+"  "+IntToStr(c->nas));
        c=c->next; //Переходим к следующему элементу очереди
  }
```

```
}
//---------------------------------------------------------------------------
//Кнопка или пункт меню "Добавить"
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  add(Edit1->Text, StrToInt(Edit2->Text));
}
//---------------------------------------------------------------------------
//Кнопка или пункт меню "Удалить"
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  del( );       //Вызов подпрограммы del для удаления первого элемента
  print( );     //Вывод изменённой очереди в Memo1
}
//---------------------------------------------------------------------------
//Кнопка или пункт меню "Просмотр"
void __fastcall TForm1::Button3Click(TObject *Sender)
{
  print( );
}
//---------------------------------------------------------------------------
//Кнопка или пункт меню "Решение"
//Удаление из очереди всех городов до города с максимальным населением
void __fastcall TForm1::Button5Click(TObject *Sender)
{
    //Если очередь пустая, то выводится об этом сообщение и выполнение
     //прерывается.
   if (first==0){
         ShowMessage("Очередь пустая");
         return;}
  //Поиск города с максимальным населением
 //Начальное значение максимального населения – население первого горо-
да
 int max=first->nas;
  //Начальное значение города с максимальным населением – название перво-
го
  //города
 AnsiString maxgor=first->gorod;
 Elem *c=first;
 while (c!=0){
         if (c->nas>max)    //Если население очередного города больше
                            //максимального,
                 {max=c->nas; //запоминаем в max население этого города
                 maxgor=c->gorod;}  // и его название
```

```
    c=c->next;
  }
  Edit3->Text=maxgor;
  //Удаление из очереди всех городов до города с максимальным населением
  c=first;                          //Становимся на первый элемент очереди
  while (c->gorod!=maxgor){   //Пока не дошли до города с максимальным
        del( );                     //населением, удаляем первый элемент очере-
ди

  }
}
//---------------------------------------------------------------------------
//Кнопка или пункт меню "Очистка"
void __fastcall TForm1::Button6Click(TObject *Sender)
{
  //Очистка памяти: пока есть первый элемент, удалять его.
  while (first!=0) del( );
  //Очистка формы
  Memo1->Clear( );
  Edit1->Clear( );
  Edit2->Clear( );
  Edit3->Clear( );
}
```

Рассмотрим пример программы со списком.

Пример 1.3

*Заполнить матрицу 4*x*5 случайными числами. Создать список из элементов матрицы, модуль которых не превышает 5. Вычислить среднее арифметическое положительных элементов списка. Удалить из списка элементы, меньшие -1. Добавить в список новый элемент после элемента с указанным значением.*

На рис. 1.4 представлено окно формы после ввода значения нового элемента (100) и элемента, после которого нужно вставить новый элемент (5) и нажатия на кнопку «Расчёт».

В начале программы объявляем тип «элемент списка»:

```
struct Element{
        int d;              //целое число
        Element *next;      //указатель на следующий элемент
};
```

Далее в программе размещаем тексты следующих подпрограмм:

Функция создания первого элемента списка (когда списка ещё нет). Параметр функции – целое число, которое будет записано в новый элемент списка. Возвращаемое значение – указатель на первый элемент списка.

```
Element *fir(int x){
   Element *c=new Element;    //Выделяем память под первый элемент
```

```
    c->d=x;          //Записываем в числовое поле элемента списка число x
    c->next=0;       //Указателю на следующий элемент присваиваем 0,
                     //так как следующего элемента ещё нет
    return c;        //Возвращаем в точку вызова созданный элемент c
}
```

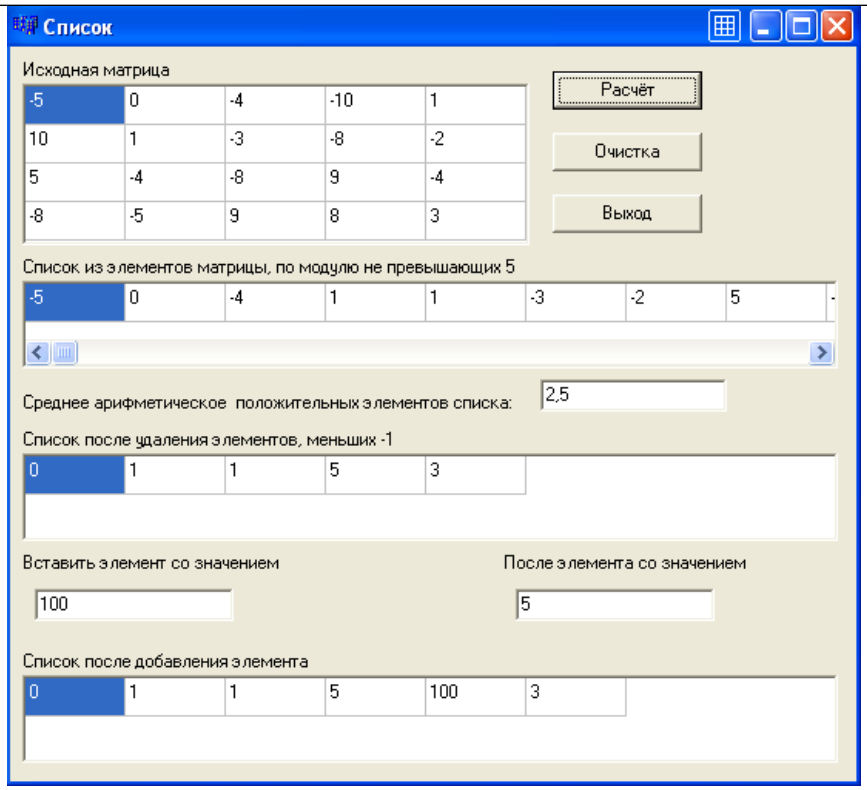


Рис. 1.4

Функция добавления элемента в конец списка. Функция не возвращает никакого значения (т. е. имеет тип void). Параметры функции – указатель на первый элемент, указатель на указатель на последний элемент и число, которое будет записано в новый элемент. Первый и последний параметры функция будет использовать, но не будет изменять, а вот указатель на последний элемент списка при добавлении нового элемента в конец обязательно изменится (последним теперь станет новый элемент). Поэтому функция должна иметь доступ для его изменения. Для этого он должен быть передан ей по адресу. Указатель на последний элемент имеет тип Element *. Значит адрес указателя на последний элемент имеет тип указатель на Element * , т. е. Element **.

```
void add(Element* fir, Element ** las, int x){
  Element *c=new Element;  //Выделяем место в памяти под новый элемент
  c->d=x;          //Записываем туда число x
  c->next=0;       //Указатель на следующий 0, так как следующего пока
нет
  (*las)->next=c;  //Следующим после последнего становится новый эле-
мент
  (*las)=c;        //Последним элементом становится новый элемент
}
```

Так как las – это указатель на указатель на последний элемент, то пользуемся операцией разадресации *las для обращения к указателю на последний.

Функция вывода списка в сетку. Параметры функции – указатель на последний элемент списка и сетка, в которую список выводится.

```
void setka(Element *fir, TStringGrid *sg){
  Element *c=fir;
  int n=0;              //n – номер заполняемой ячейки сетки: 0, 1, 2 и т.д.
  while(c!=0){
        sg->Cells[n][0]=c->d;   //Заносим число из элемента в очередную ячейку сетки
        n++;                    //и увеличиваем номер ячейки.
        c=c->next;              //Переходим к следующему элементу списка
  }
sg->ColCount=n;
}
```

Функция вычисления среднего арифметического положительных элементов списка. Возвращаемое значение – вещественное число. Параметр – указатель на начало списка (первый элемент).

```
float sr_ar(Element * fir){
  int sum=0, kol=0;
  Element *c=fir;
  while(c!=0){
        if(c->d>0){  //Если числовое значение элемента списка положительное,
              sum+=c->d;//то добавляем его к сумме
              kol++;
        }
        c=c->next;
  }
  return 1.0*sum/kol;
}
```

Функция удаления из списка элементов с числовым значением меньше -1. Параметр функции – указатель на указатель на первый элемент. В процессе удаления элементов может быть удалён и первый элемент. Аналогично функции добавления, нужно писать две «звёздочки», чтобы обеспечить доступ для изменения указателя на первый элемент списка.

Первый цикл будет удалять первые элементы, которые меньше -1, пока список не закончится либо пока первый элемент не будет больше или равен -1.

Для удаления элементов из середины списка воспользуемся вспомогательным указателем на элемент списка ***p***. Он будет хранить адрес элемента, после которого мы будем удалять элемент.

```
void udalenie(Element ** fir){
  Element *p, *c=*fir;
  //Пока значение первого элемента меньше -1 и список не закончился,
  while ((*fir)->d<-1 && (*fir)!=0){
        (*fir)=(*fir)->next;          //переходим к следующему элементу,
```

```
            delete c;           //освобождаем память из-под первого элемента,
            c=(*fir);           //и снова становимся на первый элемент
        }
        p=*fir; //Предыдущий удаляемому становится равным первому (>= -1)
        c=(*fir)->next;    //Элемент c становится равным второму элементу
        while (c!=0){
            if (c->d<-1) {
//соединяем предшествующий удаляемому со следующим после удаляемого
                p->next=c->next;
                delete c;
            //становимся на элемент, перед которым только что удалили
                c=p->next;
            }
            else{
                c=c->next;
                p=p->next;
            }
        }
    }
```

Функция очистки памяти из-под списка. Параметр функции – указатель на первый элемент списка.

```
    void o4istka(Element *fir){
      Element *c=fir;
      while (fir!=0){
            fir=fir->next;
            delete c;
            c=fir;
      }
    }
```

Функция добавления нового элемента. Параметры функции – указатель на указатель на первый элемент списка, число ***x***, которое нужно вставить, и число ***posle***, после которого выполняется вставка.

Сначала ищем элемент, после которого нужно добавить новый. Для этого используем указатель ***c1***. Цикл выполняется, пока не найден элемент со значением ***posle*** или пока не закончится список. Затем новый элемент создаётся и связывается сначала новый элемент с тем, который стоял после ***c1***, а затем ***c1*** связывается с новым элементом.

```
    void vstavka(Element ** fir, int x, int posle){
      Element *c1=(*fir);
      while (c1->next!=0 && c1->d!=posle)
            c1=c1->next;
    //Если элемент posle в списке не найден, то выводится сообщение и выходим
      if (c1->d!=posle)
            {ShowMessage ("Элемента в списке нет");
```

```
        return;}
      //Иначе добавляем элемент c после элемента c1.
  Element *c=new Element;
  c->d=x;
  c->next=c1->next;
  c1->next=c;
}
```

Далее заполняем функции обработки событий:

Функция создания формы. Первая сетка (двумерная) заполняется случайными числами из промежутка [-10, 10].

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
randomize( );              //запускаем генератор случайных чисел
for (int i=0; i<4; i++)
      for (int j=0; j<5; j++)          //присваиваем случайные числа
            StringGrid1->Cells[j][i]=IntToStr(random(21)-10);
}
```

Кнопка или пункт меню «Расчёт»

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{ Element *first=0, *last=0;
int i,j,k=0,sum=0;
for (i=0; i<4;i++)
      for (j=0; j<5; j++)
//если модуль числа меньше 5, то его нужно добавить к списку
            if(abs(StrToInt(StringGrid1->Cells[j][i]))<=5){
            if (first==0) //если списка ещё нет,
                  {first=fir(StrToInt(StringGrid1->Cells[j][i]));
            //то вызываем функцию создания первого элемента списка
                  last=first;}  //последний равен первому
            else          //иначе (если список уже есть)
            //вызываем функцию добавления очередного элемента к списку
                  add(first, &last, StrToInt(StringGrid1->Cells[j][i]));

}
  setka(first, StringGrid2);      //выводим созданный список во вторую сетку
  Edit1->Text=FloatToStr(sr_ar(first));
  udalenie(&first);         //вызываем функцию удаления элементов из списка
  setka(first, StringGrid3);      //выводим изменённый список в третью сетку
  o4istka(first);           //вызываем очистку памяти из-под списка
}
```

Лабораторная работа № 1

Динамические структуры данных. Списки, очереди, стеки

Цель работы: Изучить структуры данных список, очередь, стек и основные принципы работы с ними.

Контрольные вопросы

1 Что такое список?

2 Что такое стек?

3 Что такое очередь?

4 Какую информацию хранит элемент списка?

5 Верно ли, что элементы списка располагаются в памяти в соседних ячейках?

6 Как обратиться к элементу со значением 0 в рассмотренном ниже списке?

7 Предположим, что в процессе создания списка не запоминался указатель на последний элемент. Как его определить? Напишите фрагмент программы.

8 Что в списке на рис 1.1 обозначают представленные выражения? Если это возможно, укажите их значения:

a first->next->d

b first->next->next->next

c first->next->next->d

9 Определите, на какой элемент в списке на рис 1.1 указывает указатель c, если:

a c->d=0

b c->next->next->d=0

c c->next=first->next->next->next->next

d c->next->next->next=0

10 Предложите алгоритм и программу удаления элемента из середины:

a очереди

b стека.

Подсказка: «мешающие» элементы нужно на время перенести во вспомогательный список.

Лабораторное задание

Индивидуальные задания среднего уровня сложности

Составить схемы алгоритмов и программу для решения следующей задачи:

Создать динамическую структуру согласно варианту в табл. Л.1.1. В программе должны быть предусмотрены следующие кнопки:

«Добавить элемент»;

«Удалить элемент»;

«Просмотр»;

«Решение»;

«Очистка»;

«Выход».

При нажатии на кнопку «Решение» должны выполняться вычисления согласно заданию, указанному в табл. Л.1.1.

Должны быть предусмотрены аварийные ситуации (например: нельзя удалить элемент, если стек пустой).

В табл. Л.1.1 даны варианты индивидуальных заданий: номер варианта, название динамической структуры данных, содержимое полей данных элемента динамической структуры и задание для вычисления.

Таблица Л.1.1 − Варианты индивидуальных заданий

№ вар	Динами-ческая структура	Содержимое информационных полей	Задание для вычисления
1	Стек	Время отправления поездов: часы и минуты	Определить количество поездов, отправляющихся после 17 часов
2	Очередь	Время поезда в пути: часы и минуты	Определить максимальное время в пути
3	Стек	Погода: температура воздуха и направление ветра	Определить, сколько дней температура была ниже средней и дул северный ветер
4	Очередь	Горы: название горы и её высота	Определить самую высокую гору
5	Стек	Процессоры: название и частота	Определить процессор с минимальной частотой
6	Очередь	Экзаменационная ведомость: фамилия, оценка	Определить студентов, не сдавших экзамен
7	Стек	Мобильные телефоны: название модели и цена	Определить количество телефонов дороже среднего
8	Очередь	Мониторы: название и диагональ	Определить, есть ли монитор с диагональю 19 дюймов
9	Стек	Файлы: название и размер в байтах	Определить количество файлов с размером менее 2 Кб
10	Очередь	Товары: название товара и его цена	Определить суммарную стоимость товаров дороже 100 грн.
11	Стек	Переменные в программе: название переменной и её тип	Определить, есть ли переменные, начинающиеся на букву “а”
12	Очередь	Лекарства: название лекарства и название болезни, которая лечится этим лекарством	Определить, какими лекарствами лечится ангина
13	Стек	Изучаемые в институте дисциплины: название дисциплины и номер курса, на котором она читается	Определить все предметы, которые читаются на первом курсе
14	Очередь	Принтеры: название и скорость печати	Определить среднюю скорость печати принтеров HP

Окончание таблицы Л.1.1

№ вар	Динамическая структура	Содержимое информационных полей	Задание для вычисления
15	Стек	Футбольный клуб: название и место в чемпионате	Определить место в чемпионате введённого клуба
16	Очередь	Время отправления поездов: часы и минуты	Определить, есть ли поезда, отправляющиеся с 19 до 20 часов
17	Стек	Время поезда в пути: часы и минуты	Определить среднее время в пути
18	Очередь	Погода: температура воздуха и осадки (дождь, снег или пусто)	Определить, был ли мокрый снег (температура положительная и снег)
19	Стек	Горы: название горы и её высота	Определить количество гор выше 2000 м
20	Очередь	Процессоры: название и частота	Определить максимальную частоту процессоров Celeron
21	Стек	Экзаменационная ведомость: фамилия, оценка	Определить средний балл студентов, сдавших экзамен
22	Очередь	Мобильные телефоны: название модели (первое слово в названии – фирма-изготовитель) и цена	Сделать скидку 5 % на телефоны фирмы Nokia
23	Стек	Мониторы: название и диагональ	Определить все мониторы с диагональю больше 17 дюймов
24	Очередь	Файлы: название и тип (расширение)	Определить количество exe файлов
25	Стек	Товары: название товара и его цена	Сделать скидку 10 % на самый дорогой товар
26	Очередь	Переменные в программе: название переменной и её тип	Определить количество целочисленных переменных
27	Стек	Лекарства: название лекарства и название болезни, которая лечится этим лекарством	Определить лекарства от гриппа
28	Очередь	Изучаемые в институте дисциплины: название дисциплины и номер курса, на котором она читается	Определить количество предметов на каждом курсе
29	Стек	Принтеры: название и качество печати	Определить самый качественный принтер (если их несколько, то вывести все)
30	Очередь	Футбольный клуб: название и место в чемпионате	Определить клуб-чемпион

Индивидуальные задания высокого уровня сложности

Составить схемы алгоритмов и программы для решения задачи, указанной в табл. Л.1.2 согласно индивидуальному варианту.

Таблица Л.1.2 – Варианты индивидуальных заданий

№ вар	Задание
1	Заполнить матрицу 5 х 4 целыми числами. Создать список из отрицательных элементов матрицы. Определить наибольшее значение элементов списка. Удалить из списка элемент, расположенный после элемента со значением -3
2	Заполнить матрицу 4 х 6 вещественными числами. Создать список из элементов матрицы, попадающих в промежуток [-7, -3]. Определить произведение элементов списка, которые не делятся на 3. Вставить новый элемент после элемента со значением -5
3	Заполнить матрицу 5 х 4 целыми числами. Создать список из однозначных элементов матрицы. Определить наибольшее по модулю значение элементов списка. Удалить из списка все элементы со значением 0
4	Заполнить матрицу 6 х 4 вещественными числами. Создать список из элементов матрицы, не попадающих в промежуток [-3, 5). Определить сумму отрицательных элементов. Вставить элемент со значением 100 после каждого элемента со значением 6 или -7
5	Заполнить матрицу 5 х 4 целыми числами. Создать список из двузначных элементов матрицы. Определить наименьшее значение элементов списка. Удалить из списка элемент, расположенный до элемента со значением 25
6	Заполнить матрицу 4 х 6 вещественными числами. Создать список из элементов матрицы, квадрат которых – однозначное число. Определить произведение ненулевых элементов списка. Вставить новый элемент перед элементом о значением 1
7	Заполнить матрицу 5 х 4 целыми числами. Создать список из элементов матрицы, модуль которых не превышает 13. Определить количество отрицательных элементов списка. Удалить из списка все элементы со значением 1 или -2
8	Заполнить матрицу 6 х 4 целыми числами. Создать список из элементов матрицы, модуль которых больше 4. Определить номер минимального элемента. Вставить элемент со значением 1 после каждого положительного элемента
9	Заполнить матрицу 5 х 4 вещественными числами. Создать список из элементов матрицы, модуль которых меньше 20 и не равен 5. Определить номер максимального элемента. Удалить все элементы со значением 10

Продолжение таблицы Л.1.2

№ вар	Задание
10	Заполнить матрицу 5 х 6 целыми числами. Создать список из чётных элементов матрицы. Определить номер наибольшего по модулю элемента списка. Добавить в список элементы, равные 55, после элементов со значением из промежутка (0,5)
11	Заполнить матрицу 4 х 6 вещественными числами. Создать список из элементов матрицы, квадрат которых попадает в промежуток [30, 150]. Определить сумму модулей отрицательных элементов списка. Вставить новый элемент со значением 99 после минимального
12	Заполнить матрицу 5 х 4 целыми числами. Создать список из нечётных элементов матрицы. Определить номер наименьшего по модулю элемента списка. Удалить из списка элемент, расположенный перед минимальным по модулю
13	Заполнить матрицу 4 х 6 вещественными числами. Создать список из элементов матрицы, корень из которых попадает в промежуток [3, 11]. Определить среднее арифметическое элементов списка, расположенных после максимального. Вставить новый элемент после максимального
14	Заполнить матрицу 5 х 4 целыми числами. Создать список из однозначных элементов матрицы, кратных 3. Определить количество неотрицательных элементов списка. Удалить из списка все элементы со значением, равным максимальному
15	Заполнить матрицу 6 х 4 вещественными числами. Создать список из элементов матрицы, расположенных после максимального. Определить количество элементов списка, которые больше своих соседей. Вставить элемент со значением -111 после каждого элемента, который больше своих соседей
16	Заполнить матрицу 5 х 4 целыми числами. Создать список из двузначных элементов матрицы, цифры в которых различны (т. е. не 22, не 33 и т. д.). Определить количество элементов списка, которые меньше среднего. Удалить из списка элементы, расположенные между максимальным и минимальным
17	Заполнить матрицу 5 х 6 вещественными числами. Создать список из индексов положительных элементов матрицы. Определить сумму элементов списка, расположенных между максимальным и минимальным. Вставить новые элементы (значение ввести с клавиатуры), перед всеми элементами со значением 1
18	Заполнить матрицу 5 х 4 целыми числами. Создать список из элементов матрицы, модуль которых не превышает 20 и которые не делятся на 5. Определить количество чётных отрицательных элементов списка. Удалить из списка все элементы, отличающиеся от среднего арифметического не более чем на 2

Продолжение Окончание таблицы Л.1.2

№ вар	Задание
19	Заполнить матрицу 6 х 4 вещественными числами. Создать список из элементов матрицы, модуль которых больше суммы индексов. Определить количество элементов, расположенных до минимального. Вставить элемент со значением 11 после каждого элемента, расположенного после минимального
20	Заполнить матрицу 5 х 4 целыми числами. Создать список из элементов матрицы, которые делятся на 5 или 3 и не превышают 30. Определить количество чётных положительных элементов. Удалить все элементы с введённым значением
21	Заполнить матрицу 5 х 4 целыми числами. Создать список из элементов матрицы, не кратных 2 или кратных 3. Определить наибольшее значение элементов в первой половине списка. Удалить из списка элемент, расположенный посредине. Если таких элементов 2, то удалить их оба
22	Заполнить матрицу 4 х 6 вещественными числами. Создать список из строк матрицы, в которых не более двух отрицательных чисел. Определить сумму элементов, расположенных во второй половине списка. Вставить новый элемент после элемента, расположенного в середине списка (или между двумя средними)
23	Заполнить матрицу 5 х 4 целыми числами. Создать список из нечётных однозначных элементов матрицы. Определить произведение элементов списка, расположенных между минимальным и максимальным. Удалить из списка все элементы, кратные 5
24	Заполнить матрицу 6 х 4 вещественными числами. Создать список из элементов матрицы с чётными индексами, не попадающих в промежуток [-3, 5). Определить сумму всех минимальных и максимальных элементов. Вставить новый элемент после каждого элемента со значением, большим 15
25	Заполнить матрицу 5 х 5 целыми числами. Создать список из элементов матрицы, расположенных выше главной диагонали. Определить, является ли список возрастающим (ответ вывести словом). Удалить из списка элементы, нарушающие возрастание (в каждой такой паре удалять второй элемент)
26	Заполнить матрицу 4 х 4 вещественными числами. Создать список из элементов матрицы, расположенных ниже главной диагонали. Определить минимальный элемент во второй половине списка. Вставить новый элемент перед элементами со значением, равным минимуму второй половины списка
27	Заполнить матрицу 5 х 4 целыми числами. Создать список из элементов столбцов матрицы, в которых не менее трёх положительных элементов. Определить количество элементов списка, которые делятся на свой порядковый номер. Удалить из списка все элементы, у которых оба соседа имеют иной знак

Окончание таблицы Л.1.2

№ вар	Задание
28	Заполнить матрицу 6 х 4 целыми числами. Создать список из сумм отрицательных элементов матрицы и их индексов. Определить минимальный элемент среди элементов с чётными порядковыми номерами. Вставить новый элемент после каждого элемента, который делится на свой порядковый номер
29	Заполнить матрицу 5 х 5 вещественными числами. Создать список из элементов матрицы, стоящих на обеих диагоналях. Определить элемент с максимальной дробной частью. Удалить все элементы, большие 10
30	Заполнить матрицу 5 х 6 целыми числами. Создать список из элементов матрицы, которые делятся хотя бы на один из своих индексов. Определить номер элемента с минимальной дробной частью. Добавить в список элемент, равный сумме элементов списка, с двух сторон от элемента с минимальной дробной частью

2 Текстовые файлы. Работа с датой и временем

2.1 Файлы

Файл – это набор данных, который хранится на магнитном носителе компьютера. Каждый файл имеет имя и расширение, которое указывает операционной системе на то, что это за файл и какие действия с ним допустимы.

C++ различает два типа файлов (они не связаны с какими-то конкретными типами файлов в операционной системе): **текстовые** и **бинарные** (двоичные). Эти типы условны: любой файл в пределах одной программы может использоваться
и как текстовый, и как бинарный.

Программа на C++ может выполнять с файлами те же действия, что и пользователь, работающий с файлом в операционной системе:

открывать и закрывать файлы;

записывать в файл информацию;

читать информацию, записанную в файле;

редактировать данные в файле.

Следует различать **физический файл** (файл, расположенный на жёстком диске и характеризующийся именем файла) и **файловую переменную**, посредством которой программа работает с физическим файлом. Файловая переменная имеет особый тип «указатель на файл». Она объявляется в программе так:

```
FILE * f;
```

После этого объявления переменную **f** можно использовать для обращения к файлу.

Чтобы программа могла работать с файлами типа FILE *, нужно подключить заголовочный файл stdio.h:

```
#include <stdio.h>
```

Чтобы открыть файл, нужно воспользоваться командой **fopen**. При открытии файла переменная **f** связывается с файлом на диске (заданным своим именем). Кроме того, указывается, для чего файл открывается: для записи, чтения, добавления данных в конец или комбинации этих действий.

Пример открытия файла:

```
f=fopen(“mydoc.txt”,”rt”);
```

Эта команда открывает файл mydoc.txt как текстовый (буква t) для чтения из него информации (буква r).

Буква t может быть пропущена (это тоже означает, что файл текстовый) или вместо неё может стоять буква b (это означает, что файл открывается как бинарный). Первый параметр в кавычках обозначает имя физического файла (вместо него может быть указана переменная типа «массив символов»), а второй параметр также в кавычках обозначает режим, в котором файл открывается (т. е. что с ним разрешается делать). Перечень возможных значений параметра режима открытия файла приведён в табл. 2.1.

Таблица 2.1 – Параметры режима открытия файла

Параметр	Описание
r	файл открывается для чтения (запись при этом не разрешена)
w	файл открывается для записи; если файл существует, он пересоздаётся (чтение не разрешено)
a	файл открывается для добавления данных в конец файла (чтение не разрешено)
r+	файл открывается для чтения с возможностью записи
w+	файл открывается для записи с возможностью чтения
a+	файл открывается для добавления (записи данных в конец файла) с возможностью чтения

Не всегда удаётся открыть файл. При этом сообщение об ошибке не выводится, и при попытке выполнить любые действия с неоткрытым файлом программа аварийно завершается. Чтобы этого не происходило, нужно каждый раз после команды fopen проверять, открылся ли файл:

if (f==NULL){ShowMessage (“Не удалось открыть файл”); return;}

(если указатель на файл **f** равен NULL, т. е. файл не открылся, то выдать сообщение об ошибке и прервать выполнение программы).

Проверить, открылся ли файл, можно одновременно с открытием:

```
if ((f=fopen(“mydoc.txt”,”rt”))==NULL)
        {ShowMessage (“Не удалось открыть файл”); return;}
```

2.2 Текстовые файлы

Текстовые файлы состоят из строк.

Для чтения из файла данных построчно используется команда **fgets**. Например:

fgets(s, 40, f);

Эта команда считывает из файла f в переменную s строку размером не более 40 символов (если строка больше, то считывается только 40 символов). Переменная s должна быть объявлена как С-строка:

char * s;

или:

char s[40];

Для записи строки s в файл используется команда **fputs**:

fputs(s, f);

Эта команда записывает в файл строку s. При этом переход на новую строку не выполняется.

Чтобы сохранить и закрыть файл, нужно написать команду:

fclose(f);

Последовательность работы с файлом при записи данных такова:

1. открыть или создать файл для записи (или для чтения и записи) в конец файла:

f=fopen(filename,”at+”);

или

f=fopen(filename,”wt+”);

2. записать данные в файл:
   fputs(S,f);

или

fprint(f, <данные типа (char *) с форматом либо без>);

3. закрыть файл - fclose(f);

Здесь filename – переменная (С-строка), которая содержит физическое имя файла, а S – строка, из которой в файл записываются данные.

Чтение данных из файла выполняется в такой последовательности:

1. открыть файл для чтения:
   f=fopen(filename,"rt+");
2. осуществить чтение данных из файла:
   fscanf(f, <форматированные данные>);

или

fgets(S,80,f);

3. закрыть файл:
   fclose(f);

Для чтения последовательно всех данных из файла необходимо организовать цикл пока не достигли конца файла, используя либо функцию **feof(f),** которая возвращает значение **true** при достижении конца файла:

while (!feof(f)){ fgets(s, 40, f); . . .}

либо функцию чтения данных, которая дает нулевой результат при достижении конца файла:

while (fgets(s, 40, f)) {. . .}

т. е. буквально: считывать строки файла, пока они есть.

При чтении или записи указатель текущей записи файла автоматически смещается на количество обработанных байтов. Узнать позицию указателя можно функцией **ftell**, которая возвращает текущую позицию:

long ftell(FILE *f);

Изменить позицию указателя в файле **f** можно функцией **fseek**:

int fseek(FILE *f, long offset, int whence);

Эта функция задает сдвиг ***offset*** на количество байтов относительно точки отсчета, определяемой параметром ***whence***. Параметр whence может принимать значения, указанные в табл. 2.2:

Таблица 2.2

константа	**Whence**	**Точка отсчета**
SEEK_SET	0	Начало файла
SEEK_CUR	1	Текущая пози-
SEEK_END	2	Конец файла

Если задано значение whence = 1, то ***offset*** может быть положительным (сдвиг вперед) или отрицательным (сдвиг назад). Функция **rewind** перемещает указатель на начало файла (позиция 0). Впрочем, то же самое можно сделать оператором

fseek(f, 0, 0);

2.3 Примеры программ с использованием текстовых файлов

Пример 2.1

Создать текстовый файл из строк Memo. Имя файла вводится в Edit. Определить количество строк файла длиной более 10 символов. Вывести в Memo1 строки, в которых есть "abc".

На рис. 2.1 представлено окно формы проекта после ввода имени файла, нажатия на кнопку «Принять», ввода строк файла и нажатия на кнопки «Заполнить файл», «Просмотр» и «Решение».

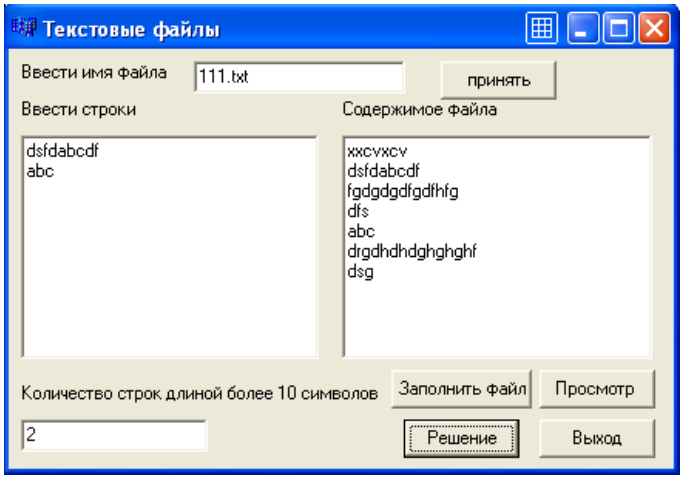


Рис. 2.1

Обратите внимание на то, что все строки типа AnsiString приходится преобразовывать командой c_str( ) в С-строку, и, наоборот, все С-строки для вывода на форму нужно преобразовывать в AnsiString.

Текст программы

```
#include <stdio.h>//для работы с файлами типа FILE*
  . . .
char *filename;//глобальная переменная – строка для имени физического
файла
FILE *f;
//---------------------------------------------------------------------------
//Кнопка «Принять»
void __fastcall TForm1::Button1Click(TObject *Sender)
{
 filename=new char [20]; //Выделяем в памяти место под строку – имя файла
 if(Edit1->Text!="")
    strcpy(filename,Edit1->Text.c_str( ));//Копируем из Edit1 в filename имя
файла
 else ShowMessage("Не введено имя файла");
}
//---------------------------------------------------------------------------
//Кнопка «Заполнить файл»
void __fastcall TForm1::Button2Click(TObject *Sender)
{
//Открываем файл для записи и выполняем проверку, удалось ли его открыть
f=fopen(filename,"w+t");
```

```
if (f==0)
        {ShowMessage("Не удаётся создать файл "); return;}
char s[40];           //строка для записи данных в файл
int n=Memo1->Lines->Count;
for (int i=0; i<n;i++){
        //Копируем очередную строку из Memo1 в s, преобразуя её в С-строку
        strcpy(s,Memo1->Lines->Strings[i].c_str( ))
        strcat(s,"\n");              //Добавляем к ней символ перевода курсора
                        //(иначе в файле всё будет расположено в одну строку)
        puts(s, f);             //Записываем строку s в файл
}
fclose(f);     //Закрываем файл (при этом он сохраняется)
}
//---------------------------------------------------------------------------
//Кнопка «Просмотр»
void __fastcall TForm1::Button3Click(TObject *Sender)
{
if((f=fopen(filename,"r+t"))==0) //Открываем файл для чтения и проверяем
        {ShowMessage("Не удаётся открыть файл "); return;}
char s[40];
//Считываем из файла строки, пока «есть что считывать»,
//т. е. не дошли до конца файла
while(fgets(s,40,f))
{
//Если последний символ строки – символ перевода курсора
//(мы его добавляли при создании), то удаляем из строки последний символ
        if (s[strlen(s)-1]=='\n') s[strlen(s)-1]='\0';
        Memo2->Lines->Add((AnsiString) s);
}
fclose(f);
}
//---------------------------------------------------------------------------
//Кнопка «Решение»
void __fastcall TForm1::Button4Click(TObject *Sender)
{int k=0;
if((f=fopen(filename,"r+t"))==0)
       {ShowMessage("Не удаётся открыть файл "); return;}
char s[40];
Memo1->Clear( );
while(fgets(s,40,f))
{
       if (s[strlen(s)-1]=='\n') s[strlen(s)-1]='\0';
       if(strlen(s)>10)k++;
       if(strstr(s,"abc"))Memo1->Lines->Add((AnsiString) s);
}
Edit2->Text=IntToStr(k);
```

```
fclose(f);
}
```

Пример 2.2

Ввести данные типа строка в компонент Memo, записать их в файл, прочитать данные из файла и отобразить их в другом Memo. Реализовать следующие операции:

- ✓ *вычислить и вывести длину каждой строки;*
- ✓ *определить упорядочены ли строки по возрастанию (каждая последующая строка больше предыдущей по алфавиту);*
- ✓ *подсчитать количество строк в файле, в которых встречается хотя бы один пробел;*
- ✓ *образовать сумму всех встречающихся чисел (только положительных, целых);*
- ✓ *предоставить пользователю возможность очистить файл (удалить содержимое файла).*

Возможный вариант окна рабочей формы изображён на рис. 2.2.

Рис. 2.2

Текст программы

```
FILE *f;
//Кнопка «Записать» (Записать данные в файл)
void __fastcall TForm1::Button1Click(TObject *Sender)
{
char s[40];
f=fopen(fn,"wt");
if (f==0) {ShowMessage("Не открыт!");return;}
int n=Memo1->Lines->Count;
for (int i=0;i<n;i++)
{       if (Memo1->Lines->Strings[i]=="") continue;//Пропускаем пустые строки
        strcpy(s,Memo1->Lines->Strings[i].c_str( ));
        strcat(s,"\n");
        fputs(s,f);
}
fclose(f);}
//Кнопка «Просмотр» (Просмотреть содержимое файла)
```

```
void __fastcall TForm1::Button4Click(TObject *Sender)
{
char s[40];
Memo2->Clear( );
f=fopen(fn,"rt+");
if (f==0) {ShowMessage("Не открыт!");return;}
while (fgets(s,40,f))
{
        if (s[strlen(s)-1]=='\n') s[strlen(s)-1]='\0';
        Memo2->Lines->Add((AnsiString) s);
}
fclose(f);
}
//---------------------------------------------------------------------------
```

//Кнопка «Обнулить» *(Очистить файл)*

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
f=fopen(fn,"wt");
if (f==0) {ShowMessage("Не открыт!");return;}
fclose(f);
}
//---------------------------------------------------------------------------
```

//Кнопка «Решение1» *(Вычислить длину каждой строки в файле)*

```
void __fastcall TForm1::Button3Click(TObject *Sender)
{
char s[40];
int dl;
Memo1->Clear( );
f=fopen(fn,"rt+");
if (f==0) {ShowMessage("Не открыт!");return;}
while (fgets(s,40,f))
{
        if (s[strlen(s)-1]=='\n') s[strlen(s)-1]='\0';
        dl=strlen(s) ;
        Memo1->Lines->Add(IntToStr(dl));
}
fclose(f);
}
//---------------------------------------------------------------------------
```

//Кнопка «Решение2»

//Расположены ли строки в файле по возрастанию их длины?

```
void __fastcall TForm1::Button5Click(TObject *Sender)
{
char s[40];
int dl1,dl2;
```

```
f=fopen(fn,"rt+");
if (f==0) {ShowMessage("Не открыт!");return;}
fgets(s,40,f);
if (ftell(f)>0)
{
        if (s[strlen(s)-1]=='\n') s[strlen(s)-1]='\0';
        dl1=strlen(s);
        int flag=0;
        while (fgets(s,40,f))
        {
                if (s[strlen(s)-1]=='\n') s[strlen(s)-1]='\0';
                dl2=strlen(s);
                if (dl1>dl2){flag=1;break;}
                dl1=dl2;
        }
        if (flag==0)ShowMessage("Строки упорядочены по возрастанию!");
        else ShowMessage("Строки не упорядочены!");
}
fclose(f);
}
//---------------------------------------------------------------------------
```

//Кнопка «Решение3»

//Количество строк в файле, содержащих хотя бы один пробел

```
void __fastcall TForm1::Button6Click(TObject *Sender)
{
char s[40];
int kol=0;
char *p;
f=fopen(fn,"rt+");
if (f==0) {ShowMessage("Не открыт!");return;}
while (fgets(s,40,f))
{
        if (s[strlen(s)-1]=='\n') s[strlen(s)-1]='\0';
        p=strchr(s,' ');
        if (p!=0) kol++;
}
ShowMessage("Количество строк, имеющих пробелы, \n\tравно " +
                        IntToStr(kol));
fclose(f);
}
//---------------------------------------------------------------------------
```

//Кнопка «Решение4» *(сумма цифр в файле)*

```
void __fastcall TForm1::Button7Click(TObject *Sender)
{
char s[40];
```

```
char c[20],*p;
int sum=0;
int m;
f=fopen(fn,"rt+");
if (f==0) {ShowMessage("Не открыт!");return;}
while (fgets(s,40,f))
{
        if (s[strlen(s)-1]=='\n') s[strlen(s)-1]='\0';
        for (int i=0;i<strlen(s);i++)
        {       m=0;
                if (s[i]>='0'&&s[i]<='9')
                        {c[m]=s[i];m++;i++;
                        while (s[i]>='0'&&s[i]<='9')
                                {c[m]=s[i];i++;m++;}
                        c[m]='\0';
                        p=&c[0];
                        sum+=StrToInt(p);}
        }
}
ShowMessage("Сумма цифр \nравна "+IntToStr(sum));
fclose(f);
}
```

Пример 2.3

Ввести сведения о товарах в текстовый файл: название товара, цена товара, количество товара на складе. Прочитать данные из файла и вывести все сведения о товарах, количество которых меньше 10.

На рис. 2.3 изображён возможный вид окна формы проекта.

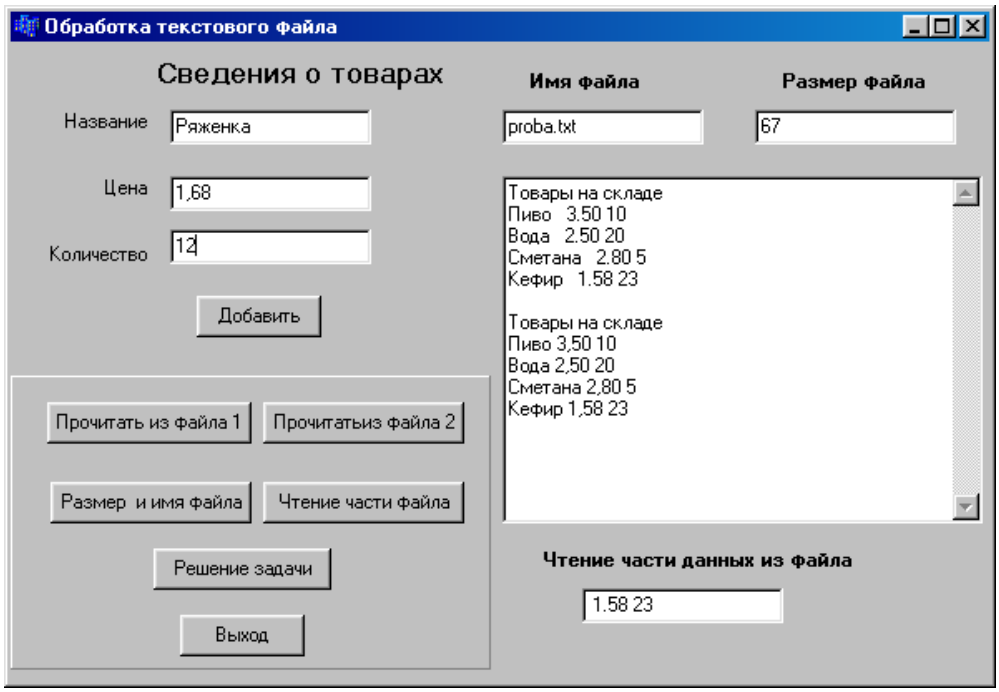


Рис. 2.3

Текст программы

```
#include <stdio.h>
// Глобальное объявление файловой переменной и определение имени файла
FILE *f;
char s[]="proba.txt";
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{///Проверяется наличие файла на диске: если файл есть, то он открывается
//для чтения и записи, а если его нет, создается для чтения и записи
if ( FileExists(s) )
        f = fopen(s,"rt+");
else
        f = fopen(s,"wt+");
 fclose(f);
 }
//---------------------------------------------------------------------------
//Кнопка "Добавить"
//добавление данных осуществляется в конец файла
void __fastcall TForm1::Button1Click(TObject *Sender)
{
if ((f=fopen(s,"at"))==NULL)
        {ShowMessage("Файл не удается открыть");
        return;}
char N[80];
```

```
strcpy(N,Edit1->Text.c_str( ));
float price;
price=StrToFloat(Edit2->Text);
int kol;
kol=StrToInt(Edit3->Text);
fseek(f, 0, SEEK_END);          //возможный вариант перемещения
                        //указателя файла в конец файла для добавления данных
fprintf(f,"%s %6.2f %i\n",&N,price,kol);
fclose(f);
Edit1->Clear( ); Edit2->Clear( );Edit3->Clear( );
Edit1->SetFocus( );
}
//---------------------------------------------------------------------------
//Функция определения размера файла
long filesize(FILE *stream)
{ long curpos, length;
  curpos = ftell(stream);
  fseek(stream, 0L, SEEK_END);
  length = ftell(stream);
  fseek(stream, curpos, SEEK_SET);
  return length;}
//---------------------------------------------------------------------------
//Кнопка "Прочитать из файла 1"
//Здесь используется проверка чтения всех байтов данных в файле
//Чтение данных из файла осуществляется по строкам
//С помощью функции filesize(FILE *stream) определяется размер файла
//Контроль достижения конца файла осуществляется с помощью
//последовательного определения смещения указателя файла на количество
//байтов для текущей строки.
void __fastcall TForm1::Button3Click(TObject *Sender)
 {
  if ((f=fopen(s,"rt+"))==NULL)
        {ShowMessage("Файл не удается открыть");
        return;}
  char N[80];
  int k   =  filesize(f);  //определяем общее количество байтов в файле
  Memo1->Clear( );
  Memo1->Lines->Add("Товары на складе");
  int m;
  m=ftell(f); //определяем количество байтов, прочитанных на данный момент
  while (m<k)
  {
        fgets(N,80,f);
        if (N[strlen(N)-1]=='\n')N[strlen(N)-1]=0;
                        Memo1->Lines->Add (AnsiString(N));
```

```
        m = ftell(f);
    }
  fclose(f);
}
//---------------------------------------------------------------------------
```
//Кнопка "Прочитать из файла 2"
```
//Вывод данных осуществляется по формату, а контроль достижения конца
//файла функцией feof(f)
//Следует приглядеться к результатам вывода данных по кнопке
//«Прочитать из файла 1»  (в Мето первая часть данных) и по кнопке
//«Прочитать из файла 2» (вторая часть данных). Данные отличаются
//отображением символа разделителя дробной части
void __fastcall TForm1::Button6Click(TObject *Sender)
{
  AnsiString  ss;
  float price; int kol;
  if ((f=fopen(s,"rt+"))==NULL)
        {ShowMessage("Файл не удается открыть"); return;}
  char N[80];
  Memo1->Lines->Add("");
  Memo1->Lines->Add("Товары на складе");
  while (!feof(f)){
        fscanf(f,"%s%f%i\n",&N,&price,&kol);
        Memo1->Lines->Add(AnsiString(N)+" "+FloatToStrF(price,ffFixed,6,2)+
                          " "+IntToStr(kol));
   }
  fclose(f);
}
//---------------------------------------------------------------------------
```
//Кнопка «Решение задачи»
```
void __fastcall TForm1::Button4Click(TObject *Sender)
{
   if ((f=fopen(s,"rt"))==NULL)
        {ShowMessage("Файл не удается открыть");
        return;}
   char N[20];  float price; int kol;
   Memo1->Lines->Add("");
   Memo1->Lines->Add("Товары, количество которых меньше 10");
   do  {
        fscanf(f,"%s%f%i",&N,&price,&kol);
        if (kol<10)
        Memo1->Lines->Add(AnsiString(N)+"
       "+FloatToStrF(price,ffFixed,6,2)+ " "+IntToStr(kol));
    } while(!feof(f));
  fclose(f);
```

```
}
//---------------------------------------------------------------------------
//Кнопка «Размер и имя файла»
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  long length;
  if ((f=fopen(s,"rt"))==NULL)
       {ShowMessage("Файл не удается открыть");
       return;}
  fseek(f, 0, SEEK_END);
  length = ftell(f);
  fclose(f);
  Edit4->Text=s;
  Edit5->Text=IntToStr(length);
}
//---------------------------------------------------------------------------
//Кнопка «Чтение части файла»
//Для чтения данных от конца файла необходимо переместиться
//на конец файла, а затем подняться на требуемое количество байт
//(знак минус)
void __fastcall TForm1::Button7Click(TObject *Sender)
{
char st[80];
if ((f=fopen(s,"rt"))==NULL)
       {ShowMessage("Файл не удается открыть");
       return;}
fseek(f, -10,2);
fgets(st,80,f);
if (st[strlen(st)-1]=='\n')st[strlen(st)-1]=0;
Edit6->Text=st;
fclose(f);
}
```

2.4 Работа с датой и временем.

В табл. 2.3 представлен перечень функций преобразования дат и времени в строку и наоборот.

Таблица 2.3 − Функции преобразования дат и времени

Название функции	Синтаксис / описание
DateTimeToStr	AnsiString DateTimeToStr(TdateTime DateTime) Преобразует DateTime в строку
DateTimeToString	Преобразует DateTime в строку по формату
FormatDateTime	Преобразует DateTime в строку по формату
DateToStr	Преобразует Date в строку
TimeToStr	Преобразует Time в строку

StrToDate	Преобразует AnsyString строку в дату TDateTime
StrToTime	Преобразует AnsyString строку во время TDateTime
StrToDateTime	Преобразует AnsyString строку в дату и время TDateTime
DayofWeek	int DayofWeek(TdateTime Date) извлекает из Date день недели (от 1 до 7, 1– воскресенье)
DecodeDate	void DecodeDate(TdateTime Date Date Word &Year, Word &Month,Word&Day) – разбивает Date на год, месяц, день)
DecodeTime	void DecodeTime(TdateTime Time Word &Hour, Word &Min, Word &Sec, Word &MSec) – разбивает Time на часы, минуты, секунды, миллисекунды
EncodeDate	TDateTime EncodeDate(Word Year, Word Month,Word Day) – преобразует год, месяц, день в TDateTime
EncodeTime	TDateTime EncodeTime(Word Hour, Word Min, Word Sec,Word MSec) – преобразует часы, минуты, секунды и миллисекунды в TDateTime

В функциях **DateTimeToString** и **FormatDateTime**, которые представлены в табл. 2.3, используется строка форматирования дат. В табл. 2.4 приведены значения спецификаторов формата дат.

Таблица 2.4 − Спецификаторы формата дат

Спецификатор	Описание
C	Вывод даты и времени по заданной глобально переменной ShortDateFormat
D	Вывод дня без начальных нулей (1-31).
dd	Вывод дня с начальным нулём (01-31)
ddd	Вывод дня в виде аббревиатуры (пн-вс)
dddd	Вывод дня в виде полного названия дня недели (понедельник-воскресенье)
M	Вывод месяца без начальных нулей (1-12)
mm	Вывод месяца с начальными нулями (01-12)
mmm	Вывод месяца в виде аббревиатуры (янв-дек)
mmmm	Вывод полного названия месяца (Январь-Декабрь)
yy	Вывод двух последних цифр года (00-99)
yyyy	Вывод года в полном формате (0000-9999)
h	Вывод часа без начального нуля (0-23)
hh	Вывод часа с начальным нулём (00-23)
n	Вывод минут без начального нуля (0-59)
nn	Вывод минут с начальным нулем (00-59)
s	Вывод секунд без начального нуля (0-59)
ss	Вывод секунд с начальным нулем (00-59)
am/pm **a/p**	Использование 12-часовой шкалы и символов **am** или **pm** Использование 12-часовой шкалы и символов **a** или **p**

В табл. 2.5 представлены некоторые методы, наиболее часто используемые при работе с данными типа «дата» и «время». Обратите внимание на синтаксис при использовании методов: за переменной, к которой применяется метод, следует точка, после которой пишется имя метода.

Пример:

d.TimeString( );

Таблица 2.5 − Некоторые методы работы с датой и временем

Метод	Описание
CurrentDate( )	Возвращает текущую дату
CurrentTime( )	Возвращает текущее время
CurrentDateTime( )	Возвращает текущие дату и время
Date( )	Возвращает текущую дату
Time( )	Возвращает текущее время
Now( )	Возвращает текущие дату и время
operator int	Преобразует TDateTime в целое
operator - ; **operator +**	С их помощью можно из значения даты (времени) вычитать, складывать даты, числа типа double, числа типа int
DateString( )	Выводит (преобразует в AnsiString) дату TDateTime в строку, например в Edit
TimeString( )	Выводит (преобразует AnsiString) время TDateTime в строку, например в Edit

Пример использования **operator** для вывода в Edit1:

```
TDateTime d1,d2,d3;
d2=TDateTime("14.10.2007 14:15");
d1=AnsiString("14.10.2007 18:10");
d3=d2.operator-(d1);
Edit1->Text=d3.TimeString( );
```

В результате действия d2.operator-(d1) увидим в Edit1 значение 3:55. Здесь же показаны два способа присвоения начального значения переменным d2 и d1.

В табл. 2.6 представлены некоторые функции, наиболее часто используемые при обработке данных типа «дата» и «время».

Таблица 2.6 − Некоторые функции для обработки данных типа TdateTime

Функция/Синтаксис	Описание
int **YearsBetween** (TdateTime Now, TdateTime Athen)	Возвращает целое количество лет между датами Now и Athen
double **YearSpan** (TdateTime Now, TdateTime Athen)	Возвращает вещественное количество лет между датами Now и Athen, (с учетом части года)
int **MonthsBetween**(TdateTime Now, TdateTime Athen)	Возвращает целое количество месяцев между датами Now и Athen
double **MonthSpan**(TdateTime Now, TdateTime Athen)	Возвращает вещественное количество месяцев между датами Now и Athen, (с учётом части месяца)
int **HoursBetween**(TdateTime Now, TdateTime Athen)	Возвращает целое количество часов между Now и Athen
double **HourSpan**(TdateTime Now, TdateTime Athen)	Возвращает вещественное количество часов между Now и Athen, (с учётом части часа)
int **MinutesBetween**(TdateTime Now, TdateTime Athen)	Возвращает целое количество минут между Now и Athen
double **MinuteSpan**(TdateTime Now, TdateTime Athen)	Возвращает вещественное количество минут между Now и Athen, (с учётом части минуты)
int **SecondsBetween**(TdateTime Now, TdateTime Athen)	Возвращает целое количество секунд между Now и Athen
double **SecondSpan**(TdateTime Now, TdateTime Athen)	Возвращает вещественное количество секунд между Now и Athen, (с учётом части секунды)
int **MillisecundsBetween**(TdateTime Now, TdateTime Athen)	Возвращает целое количество миллисекунд между Now и Athen
double **MillisecundSpan**(TdateTime Now, TdateTime Athen)	Возвращает вещественное количество миллисекунд между Now и Athen, (с учётом части миллисекунды)

2.5 Примеры использования методов и функций для обработки даты и времени

Пример 2.4

Определить время в пути, если задано время отправление, время прибытия и количество суток в пути.

```
TDateTime d1,d2,d3,d4,d5;
Word Year, Month, Day1,Day2;
int sut;
unsigned short int Hour,Min,Sec,MSec;
d1=Edit1->Text;
//читать и выводить данные типа TDateTime можно без функций
//преобразований
d2=StrToTime(Edit2->Text);      //использование функции преобразования
if (Edit5->Text=="") sut=0;
else
        sut=StrToInt(Edit3->Text);
if (d2<=d1)
        {d3=d2.operator+(1)-d1;
        DecodeTime(d3, Hour, Min, Sec, MSec);
        Hour+=24*sut;  }
else {d3=d2-d1; DecodeTime(d3, Hour, Min, Sec, MSec);}
AnsiString s="";
if (Min<10) s="0"+IntToStr(Min);
else  s=IntToStr(Min);
Edit4->Text=d2;
//Edit4->Text=d2.TimeString( );
Edit4->Text=IntToStr(Hour)+":"+s;
}
```

Общее количество часов можно определить и в следующем фрагменте:

```
d2=Date( )+d2;
DecodeDate(d3, Year, Month, Day1);
d4=d2.operator+(sut);
DecodeDate(d4, Year, Month, Day2);
Hour=24*(Day2-Day1);
```

Пример 2.5

Определить текущий день и время.

```
Word Year, Month, Day, Hour, Min, Sec, MSec;
TDateTime dt = Now( );
DecodeDate(dt, Year, Month, Day);
Label1->Caption = AnsiString("Сегодня ") + IntToStr(Day) + AnsiString(" дня ")
        +IntToStr(Month)+ " месяца " + IntToStr(Year)+" года ";
```

```
DecodeTime(dt, Hour, Min, Sec, MSec);
Label2->Caption =IntToStr(Hour)+ " часов " + IntToStr(Min)+"минут" ;
```

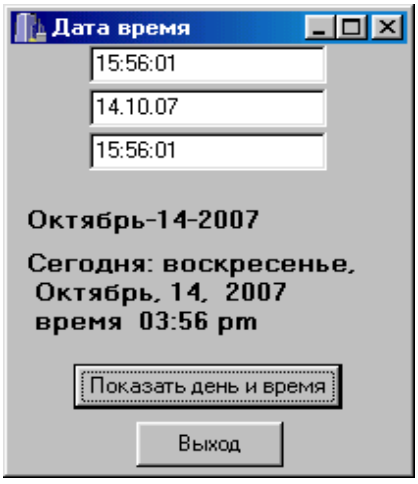


Рис. 2.4

Пример 2.6

Вывести текущую дату и время, используя различные варианты форматирования

Окно формы с результатами работы программы показано на рис. 2.4.

```
TDateTime fd;
// возможный разделитель часов минут
//TimeSeparator = '-';
Edit1->Text=fd.CurrentTime( );
fd=Date( );   Edit2->Text=fd;
fd=Time( );  Edit3->Text=fd.TimeString( );
DateSeparator = '-';
ShortDateFormat = "mmmm/dd/yyyy";
Label1->Caption = DateToStr(Date( ));
AnsiString S = "Сегодня: ";
S = S + FormatDateTime ("dddd,\n mmmm, dd, yyyy ' \n время ' hh:mm
am/pm",Now( ));
Label2->Caption=S;
```

В примере 2.6 показаны различные возможности форматирования даты и времени, в том числе возможные константы разделителей для даты и времени.

Пример 2.7

Задается время отправления транспортного средства и время в пути. Требуется определить время прибытия в конечный пункт.

```
char dlit[6];
TDateTime d1,d2;
AnsiString s1,s2;
int n;
 d1=Edit1->Text;
 strcpy(dlit,Edit2->Text.c_str( ));
 s1=(AnsiString)dlit;
n=StrToInt(s1.SubString(1,s1.Pos(':')-1));
if (n>=24) n=n%24;
s2=IntToStr(n)+":"+s1.SubString(s1.Pos(':')+1,2);
d2=d1+s2;
Edit3->Text=d2.TimeString( );
```

Пример 2.8

Вывести календарь, расположенный на странице Win32.

```
void __fastcall TForm1::DateTimePicker1CloseUp(TObject *Sender)
{
Edit1->Text=DateTimePicker1->Date;
}
```

Окно формы с результатами работы программы показано на рис. 2.5.

2.6 Пример проекта программы, использующей текстовый файл и дату/время

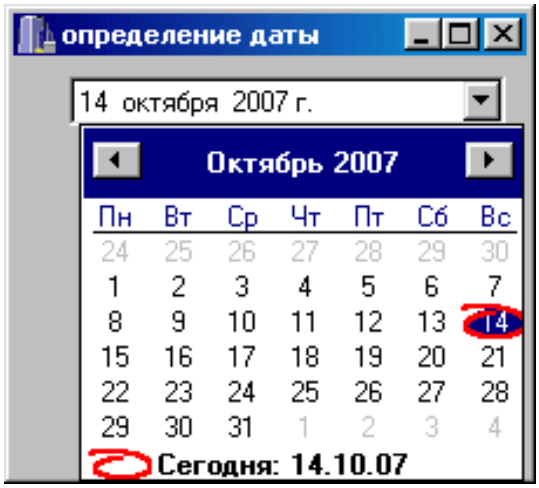


Рис. 2.5

Пример 2.9

Ввести данные по автостанции: направление маршрута, время отправления и прибытия, период действия данного расписания по направлениям, цена билета. Выполнить следующие операции:

✓ *записать данные в текстовый файл;*

✓ *распечатать эти данные с информацией о времени в пути;*

✓ *распечатать сведения о маршруте «Одесса-Киев», если это расписание будет действовать еще 45 дней относительно текущего дня;*

✓ *распечатать сведения о таком маршруте «Одесса-Варна», который прибывает в Варну ближе к полуночи, но не ранее 15:59.*

Окно формы с результатами работы программы показано на рис. 2.6.

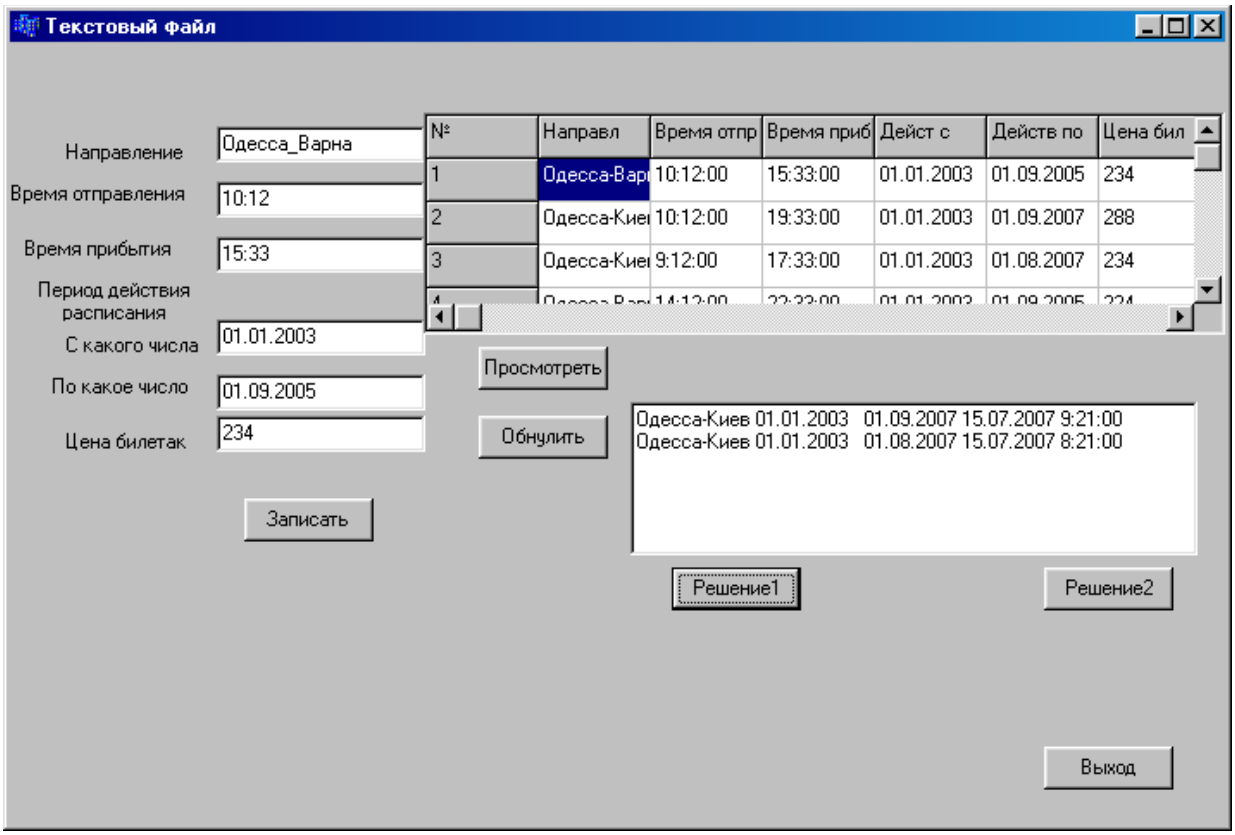


Рис. 2.6

Текст программы

```
#include <stdio.h>
//Объявление глобальных переменных
FILE *f;
char *fn="a.txt";
char npr[50];
char totp[9],tpri[9],dot[11],dpri[11];
float cena;
//---------------------------------------------------------------------------
//Кнопка «Записать» (Записать данные в файл)
void __fastcall TForm1::Button1Click(TObject *Sender)
{
f=fopen(fn,"at+");
if (f==0){ShowMessage("Не открыт!");return;}
strcpy(npr,Edit1->Text.c_str( ));
strcpy(totp,Edit2->Text.c_str( ));
strcpy(tpri,Edit3->Text.c_str( ));
strcpy(dot,Edit4->Text.c_str( ));
strcpy(dpri,Edit5->Text.c_str( ));
cena=StrToFloat(Edit6->Text);
fprintf(f,"%s %s %s %s %s %5.2f\n", &npr,&totp,&tpri,&dot,&dpri,cena);
fclose(f);
}
//---------------------------------------------------------------------------
//Кнопка «Просмотреть» (Просмотреть данные из файла)
void __fastcall TForm1::Button2Click(TObject *Sender)
{
f=fopen(fn,"rt+");
if (f==0){ShowMessage("Не открыт!");return;}
int i=1;
SG1->Cells[0][i-1]="№";
SG1->Cells[1][i-1]="Направл";
SG1->Cells[2][i-1]="Время отпр";
SG1->Cells[3][i-1]="Время приб";
SG1->Cells[4][i-1]="Действ с" ;
SG1->Cells[5][i-1]="Действ по";
SG1->Cells[6][i-1]="Цена бил";
SG1->Cells[7][i-1]="Время в пути";
SG1->RowCount=i+1;
while (fscanf(f,"%s%s%s%s%s%f", &npr,&totp,&tpri,&dot,&dpri,&cena)>0)
{
        SG1->Cells[0][i]=IntToStr(i);
        SG1->Cells[1][i]=npr;
        SG1->Cells[2][i]=TTime(totp);
```

```
        SG1->Cells[3][i]=TTime(tpri);
        SG1->Cells[4][i]=TDate(dot);
        SG1->Cells[5][i]=TDate(dpri);
        SG1->Cells[6][i]=FloatToStr(cena);
        SG1->Cells[7][i]=TTime(tpri)-TTime(totp);
        SG1->RowCount=i+1;
        i++;
}
fclose(f);
}
//---------------------------------------------------------------------------
```

//Кнопка «Обнулить»

```
void __fastcall TForm1::Button3Click(TObject *Sender)
{
f=fopen(fn,"wt");
if (f==0){ShowMessage("Не открыт!");return;}
fclose(f);
}
//---------------------------------------------------------------------------
```

//Кнопка «Решение 1» *(Определить, какие маршруты «Одесса-Киев»*
// будут действовать еще 45 дней)

```
void __fastcall TForm1::Button5Click(TObject *Sender)
{ String napr;
 TDate ToDay;
 ToDay=Date( );
 f=fopen(fn,"rt+");
 if (f==0){ShowMessage("Не открыт!");return;}
 Memo1->Clear( );
 while (fscanf(f,"%s%s%s%s%s%f", &npr,&totp,&tpri,&dot,&dpri,&cena)>0)
 {
        napr=&npr[0];
        if ((napr=="Одесса-Киев")&& ((ToDay+45)<TDate(dpri)))
                Memo1->Lines->Add(AnsiString(npr)+" "+dot+" "+" "+" "+dpri+
                        " "+(ToDay+45)+" "+(TTime(tpri)-TTime(totp)));
 }
 fclose(f);
}
//---------------------------------------------------------------------------
```

//Кнопка «Решение 2» *(Определить маршрут «Одесса-Варна», прибывающий*

```
//в Варну ближе к полуночи, но не ранее 15:59.
void __fastcall TForm1::Button6Click(TObject *Sender)
{
String napr, s;
TTime Ttmin;
```

```
Ttmin=TTime("15:59");
f=fopen(fn,"rt+");
if (f==0){ShowMessage("Не открыт!");return;}
Memo1->Clear( );
while (fscanf(f,"%s%s%s%s%s%f", &npr,&totp,&tpri,&dot,&dpri,&cena)>0)
        { napr=&npr[0];
        if (napr=="Одесса-Варна")
                if (TTime(tpri)>(Ttmin))
                        s=AnsiString(npr)+" "+totp+" "+tpri+" "+FloatToStr(cena);
                Ttmin=TTime(tpri);
        }
Memo1->Lines->Add(s);
fclose(f);
}
```

Лабораторная работа № 2
Текстовые файлы

Цель работы: Научиться создавать и обрабатывать текстовые файлы в C++.

Контрольные вопросы

1 Какими методами можно создавать текстовые файлы?

2 Как отрыть файл и сохранить изменения в нем, используя компоненты TOpenDialog и TSaveDialog?

3 Перечислите основные режимы открытия текстового файла для чтения или записи данных в конец файла.

4 Как проверить, правильно ли открыт файл?

5 Какие методы используются для чтения строкового типа данных из файла и как организовать просмотр всех данных?

6 Какова стандартная последовательность действий в программе при работе с текстовым файлом?

7 Какой заголовочный файл нужно подключить в программе, чтобы можно было работать с файлами?

8 Какого типа должна быть переменная для считывания данных из текстового файла?

9 Укажите ошибки в следующих фрагментах программ:

a)
```
f=fopen(“myf.txt”, “rt”);
fputs(s,f);
fclose(f);
```

b)
```
f=fopen(“myf.txt”, “rt”);
fgets(s, 30, f);
f=fopen(“myf.txt”, “at”);
```

c)
```
f=fopen(“myf.txt”, “rt”);
AnsiString s;
fputs(s,f);
fclose(f);
```

10 Каким будет содержимое файла myf.txt после выполнения следующего фрагмента программы:

```
f=fopen(“myf.txt”, “wt”);
fputs(“word”,f);
fputs(“after”,f);
fputs(“\n”,f);
fputs(“word”,f);
fclose(f);
```

Лабораторное задание

Составить схемы алгоритмов и программу для решения следующей задачи:

Создать текстовый файл из строк Memo.

Предоставить пользователю возможность просмотра содержимого файла.

Выполнить действия, указанные в табл. Л.2.1.

Таблица Л.2.1 − Варианты индивидуальных заданий

№ вар	Индивидуальное задание
1	Определить количество строк файла с чётной длиной. Вывести строки, которые начинаются с буквы ‘A’
2	Определить количество строк файла с длиной, не кратной 3. Вывести строки, 3которые заканчиваются цифрой
3	Определить количество строк файла, содержащих ровно одно слово. Вывести строки длиной менее 5 символов
4	Определить количество строк файла, в которых есть буква ‘w’. Вывести строки с нечётной длиной
5	Определить количество строк файла, заканчивающихся на вопросительный знак. Вывести строки, в которых есть цифры
6	Определить, есть ли в файле слово “word”. Вывести строки, которые содержат не менее 6-ти символов
7	Определить номер строки и номер позиции в этой строке, с которой начинается слово “plus”. Вывести строки, которые начинаются и заканчиваются на одну и ту же букву
8	Определить количество цифр в файле. Вывести номера строк, которые содержат не менее трёх слов
9	Определить количество строк файла, не содержащих запятых. Вывести номера строк, содержащих восклицательный знак
10	Определить общее количество знаков препинания в файле. Вывести строку (или строки) максимальной длины
11	Определить среднюю длину строк в файле. Вывести строки с длиной меньше средней
12	Определить количество слов в файле. Вывести номера строк, которые содержат букву ‘z’
13	Определить количество строк файла с длиной больше средней. Вывести строки, которые не содержат арифметических знаков
14	Определить максимальную длину строки в файле. Вывести строки, содержащие более двух слов
15	Определить количество слов файла, пишущихся через дефис. Вывести строки, которые имеют длину больше предыдущей строки
16	Определить, расположены ли строки в файле по возрастанию длины. Вывести самую короткую строку (или строки) файла
17	Определить минимальную длину строк файла. Вывести строки, которые начинаются с той же буквы, что и первая строка файла
18	Определить номер строки файла, имеющей максимальную длину. Вывести строки файла, начинающиеся с большой буквы
19	Определить количество строк файла с чётным количеством слов. Вывести первые слова строк файла
20	Определить количество пробелов в файле. Вывести строку (или строки) с наибольшим количеством пробелов

Окончание таблицы Л.2.1

№ вар	Индивидуальное задание
22	Определить, есть ли в файле слово “computer”. Вывести строки файла, содержащие большие буквы
23	Определить количество слов файла, начинающихся с буквы ‘k’. Вывести все большие буквы, содержащиеся в файле
24	Определить, есть ли в файле хотя бы два слова, начинающиеся с буквы ‘Z’. Вывести строки с наименьшим количеством слов
25	Определить, расположены ли строки файла по алфавиту. Вывести строки файла, расположенные в алфавитном порядке (т. е. те строки, которые больше предыдущей)
26	Определить суммарную длину слов файла. Вывести номера строк, не содержащих цифры
27	Определить количество слов в файле, которые заканчиваются буквой ‘y’. Вывести строки файла, удалив из них лишние пробелы
28	Определить, есть ли в файле строки, состоящие только из цифр и арифметических знаков. Вывести все числа (не цифры!) из файла в столбик в Memo
29	Определить количество чисел (не цифр!) в файле. Вычислить произведение однозначных чисел
30	Определить, сколько строк файла содержат более одного предложения. Вычислить сумму двузначных чисел, содержащихся в файле

Лабораторная работа № 3

Форматированный вывод. Работа с датой и временем

Цель работы: Научиться работать с датой и временем, а также с форматированными данными в текстовых файлах в C++.

Контрольные вопросы

1. Какими методами можно создавать текстовые файлы?
2. Перечислите основные режимы открытия текстового файла для чтения или записи данных в конец файла.
3. Приведите последовательность методов добавления данных в файл.
4. Перечислите все возможные команды, используемые для чтения и добавления данных в текстовый файл.
5. Какие типы данных используются для хранения даты и времени?
6. Перечислите известные Вам команды для работы с датой и временем.

Лабораторное задание

Подготовить протокол лабораторной работы, с указанием данных, составляющих задание, структурную схему и текст программы, а также ожидаемые результаты в виде таблицы.

Создать проект программы, позволяющий пользователю создать текстовый файл, заполнить его данными, указанными в табл. Л.3.1, и просмотреть созданный файл. Выполнить задание, указанное в табл. Л.3.1.

Таблица Л.3.1 − Варианты индивидуальных заданий

№ вар	Содержание текстового файла	Задание
1	Список принтеров для продажи: тип принтера, фирма-изготовитель, скорость работы (количество листов в минуту), цена принтера	Вывести данные о принтерах фирмы HP, которые печатают более 10 листов в минуту
2	Список абонентов телефонной станции: ФИО абонента, номер телефона, оплата за два последних месяца, сумма долга	Вывести сведения об абонентах, у которых долг за оплату превысил 200 грн.
3	Список для автоинспекции: данные об угнанных автомобилях: государственный номер, марка автомобиля, цвет, дата заявления	Вывести сведения об угнанных автомобилях марки «Opel», угнанных более двух лет назад
4	Список аварийных домов в городе: улица, номер дома, количество жителей в доме, год постановки на учет	Вывести данные о домах, которым уже более 30-ти лет аварийного состояния и при количестве более 200 жильцов в доме

Продолжение таблицы Л.3.1

№ вар	Содержание текстового файла	Задание
5	Ведомость по оплате коммунальных услугах: улица, номер дома, ФИО жильца, дата оплаты, накопившийся долг	Вывести сведения о жильцах, имеющих долг по оплате более 500 грн.
6	Список студентов в группе: номер в журнале, ФИО, оценки по математике, физике, философии	Вывести список студентов, которые не имеют троек, а также средний балл каждого студента
7	Список очередников по установке телефонов: ФИО, дата подачи заявления на установку, наличие или отсутствие льгот. Наличие или отсутствие льгот записывается по шаблону: если льготы есть, то отмечается как “есть”, иначе – “нет”	Распечатать список очередников, которые имеют льготы и которые стоят в очереди на установку более двух лет
8	Летнее расписание движения поездов: номер поезда, пункт отправления и пункт прибытия, время отправления (по шаблону: ЧЧ:ММ)	Вывести данные о поездах направления Одесса-Киев, которые отправляются в Киев между 10:00 и 17:00 часами
9	Ведомость по услугам почтовой связи: вид отправления, размер оплаты за пересылку наземным или воздушным транспортом, пункт отправления, время доставки	Вывести сведения о тех услугах, которые позволят доставить письмо на направление Полтава за три дня
10	Данные о времени интенсивной работы канала ионосферной связи: направление связи, месяц, время начала и окончания интенсивной работы канала по шаблону: ЧЧ:ММ	Вывести данные на направление Одесса – Львов в период 18:00 –23:00
11	Список линий интенсивной связи метеорного канала: направление канала, время начала и время окончания интенсивной связи (по шаблону ЧЧ:ММ:СС), месяц, для которого отмечается интенсивная связь	Вывести данные по всем направлениям, для которых длительность интенсивной связи наибольшая в ночное время, а также данные по направлению Одесса-Норильск для мая
12	Сведения о медикаментах в аптеке: наименование лекарства, срок годности, цена. Срок годности ввести по шаблону ДД.ММ.ГГГГ	Вывести данные по медикаментам, срок годности которых заканчивается в текущем году

Продолжение таблицы Л.3.1

№ вар	Содержание текстового файла	Задание
13	Данные о статистической ошибке канала телеметрии потребления электроэнергии: порядковый номер, дата (по шаблону ДД-ММ-ГГ), наименование линии направления, показания электросчетчика, показания телеметрии	Вывести данные о линиях, у которых погрешность измерений превышает 10 %, и отдельно – информацию по линии, на которой погрешность максимальна
14	Сведения о сезонном распределении температуры в городе: дата (по шаблону: ДД.ММ.РР), средняя температура за сутки	Вывести данные о днях, когда средняя температура дня была выше средней на 5 градусов
15	Составить список работников предприятия: табельный номер, ФИО, должность, дата принятия на работу, оклад.	Вывести данные о технике, получающем наибольший оклад, и данные об инженерах
16	Список фильмов в фильмотеке: порядковый номер, название фильма, время демонстрации в часах, год выпуска, год создания, количество копий	Вывести данные о фильме, срок хранения которого более 5 лет, с наибольшим временем демонстрации
17	Сведения об автомобилях для продажи: текущий номер, тип автомобиля, тип двигателя, километраж пробега, год выпуска, стартовая цена	Вывести сведения об автомобилях “Ford”, у которых пробег составляет менее 50000 км и дата выпуска не менее двухгодичной давности
18	Список средств индивидуальной защиты от поражения электрическим током: инвентарный номер, наименование, дата последней проверки (по шаблону: ДД.ММ.РР), под каким напряжением испытано	Вывести данные о средствах, испытанных напряжением более 5000 В, а также сведения обо всех средствах, испытанных последний раз более двух лет назад
19	Список сотрудников предприятия: табельный номер, ФИО, должность, год рождения, пол (по шаблону: мужчина – м, женщина – ж), адрес	Вывести сведения о среднем возрасте мужчин и женщин, и составить список женщин, которым остался год до пенсии
20	Список музыкальных дисков в звукозаписи: номер, название диска, количество песен на диске, время звучания, цена диска	Вывести данные о дисках, имеющих не менее 10-ти песен, а общее время звучания от 40 до 60 мин

Продолжение таблицы Л.3.1

№	Содержание текстового файла	Задание
21	Список ЭВМ, установленных в дисплейном зале: номер по порядку, тип ЭВМ, тактовая частота, оперативная память, емкость диска, год установки в классе	Вывести данные об ЭВМ, у которых тактовая частота ниже 1000 МГц, а оперативная память менее 256 МБ
22	Список студентов группы: номер в журнале, ФИО, рейтинг по математике, физике, информатике	Вывести список студентов, которые имеют по всем предметам рейтинг не ниже 95 баллов, а также список студентов, у которых хотя бы по одному предмету имеется 100 баллов
23	Список команды «Черноморец»: номер на поле, ФИО, амплуа (вратарь, защитник, т. д.), сколько голов забито за сезон	Вывести данные о нападающих команды, не забивших за сезон ни одного гола, и о защитниках, пропустивших больше всего голов
24	Список фильтров: номер, тип фильтра, ширина полосы пропускания, согласующее входное сопротивление фильтра	Вывести сведения о фильтрах, имеющих полосу пропускания более 5000 Кгц, а также сведения о фильтре с наименьшим входным сопротивлением
25	Список книг в книгохранилище: инвентарный номер, название книги, год издания, стоимость	Вывести сведения о книгах, названия которых начинаются со слова “Программирование”, а также список книг, которые подлежат списанию (срок хранения которых более 10-ти лет)
26	Список жильцов: улица, номер дома, номер квартиры, ФИО владельца, год последнего капитального ремонта	Вывести данные о владельцах квартир, у которых номер дома и номер квартиры совпадают, а также сведения о домах, в которых капитальный ремонт не проводился более 25 лет
27	Список транзисторов: порядковый номер, тип транзистора, материал, коэффициент передачи по току (минимальное и максимальное значения)	Вывести список транзисторов германиевого типа, а также список транзисторов, у которых коэффициент передачи по току более 45 единиц

Окончание таблицы Л.3.1

№	Содержание текстового файла	Задание
28	Список нереализованного товара на складе: наименование товара, количество, цена за единицу товара, кто купил данный товар, сумма долга за товар	Вывести общую сумму долга за товар, а также всю информацию о бетономешалках
29	Список предприятий города, которые выпускают электронное оборудование: наименование предприятия, вид выпускаемой продукции, количество за квартал, рентабельность (если предприятие рентабельно, то значение этого поля положительно, иначе – отрицательно)	Вывести сведения о предприятии, которое выпускает максимальное количество приемников, а также сведения о предприятиях, имеющих отрицательную рентабельность
30	Список абонентов телефонной станции: текущий номер абонента, ФИО, адрес, номер телефона (вводится по шаблону: ЧЧ-ЧЧ-ЧЧ), задолженность по оплате	Вывести данные об абонентах, телефонные номера которых начинаются с номера 22, а также абонентов, ФИО которых начинаются с К или Л

3 Бинарные файлы

3.1 Бинарные файлы

Бинарный (двоичный) файл представляет собой просто последовательность символов, в которой без каких-либо разделителей – пробелов, символов конца строки и т. п. – хранятся символы, отображающие самые различные объекты. Что именно и в какой последовательности лежит в бинарном файле, – должна знать программа.

Отличие бинарных файлов от текстовых заключается в том, что в них можно хранить данные определённого типа, например структуры.

С двоичными файлами можно выполнять те же действия, что и с текстовыми. Для открытия бинарного файла используется та же команда fopen, только во втором параметре (режиме открытия файла) вместо буквы ***t*** нужно указать букву ***b***. Например, следующая команда открывает бинарный файл myfile.txt для чтения:

f=fopen(“myfile.txt”, ”rb”);

Запись и чтение в двоичных файлах чаще всего производятся соответственно функциями **fwrite** и **fread.**

Функции чтения fread:

```
size_t fread(
   char *buffer,          // Массив для чтения данных
   size_t elemSize,       // Размер одного элемента
   size_t numElems,       // Число элементов для чтения
   FILE *f                // Указатель на структуру FILE
);
```

Здесь size_t определен как беззнаковый целый тип в системных заголовочных файлах. Функция пытается прочесть ***numElems*** элементов из файла, который задается указателем ***f*** на структуру ***FILE***, размер каждого элемента равен ***elemSize***. Функция возвращает реальное число прочитанных элементов, которое может быть меньше, чем numElems, в случае конца файла или ошибки чтения.

Пример использования функции fread:

```
FILE *f;
double buff[100];
size_t res;
f = fopen("tmp.dat", "rb");              //Открываем файл
if (f == 0) {
   ShowMessage("Не могу открыть файл для чтения");
   exit(1);                              //Завершить работу с кодом 1
}
                  //Пытаемся прочесть 100 вещественных чисел из
            файла
res = fread(buff, sizeof(double), 100, f);
               //res равно реальному количеству прочитанных чисел
```

В этом примере файл "tmp.dat" открывается на чтение как бинарный, из него читаются 100 вещественных чисел размером 8 байт каждое. Функция fread возвращает реальное количество прочитанных чисел, которое меньше или равно 100.

Функция fread читает информацию в виде потока байтов и в неизменном виде помещает ее в память. Следует различать текстовое представление чисел и их бинарное представление! В приведенном выше примере числа в файле должны быть записаны в **бинарном виде**, а не в виде текста. Для текстового ввода чисел следует использовать функции **ввода по формату**, которые были рассмотрены выше.

Функция бинарной записи в файл **fwrite** аналогична функции чтения fread. Она имеет следующий прототип:

```
size_t fwrite(
   char *buffer,          // Массив записываемых данных
   size_t elemSize,       // Размер одного элемента
   size_t numElems,       // Число записываемых элементов
   FILE *f                // Указатель на структуру FILE
);
```

Функция возвращает число реально записанных элементов, которое может быть меньше, чем numElems, если при записи произошла ошибка: например, не хватило свободного пространства на диске.

Пример использования функции fwrite:

```
FILE *f;
double buff[100];
size_t num;
. . .
f = fopen("tmp.res", "wb");              //Открываем файл "tmp.res"
if (f == 0) {
   ShowMessage("Не могу открыть файл для записи");
   exit(1);                              //Завершить работу программы с кодом 1
}
                        //Записываем 100 вещественных чисел в файл
res = fwrite(buff, sizeof(double), 100, f);
                     //В случае успеха res == 100
```

В обе функции передается указатель на выводимые или вводимые данные, параметр ***elemSize*** задает размер в байтах передаваемых данных, а параметр ***numElems*** определяет число передаваемых данных.

Так же, как и в текстовых, в бинарных файлах можно определять текущий указатель файла и перемещаться в произвольное место файла командами соответственно ftell и fseek. Возможность перемещать указатель особенно полезна в файлах, которые состоят из однородных записей одинакового размера. Например, если в файле записаны только действительные числа типа double, то для того, чтобы прочитать i-тое число, достаточно выполнить операторы:

```
fseek(F, sizeof(double)*(i-l), 0);
```

```
fread(&a, sizeof(double),1,F);
```

Таким образом можно читать любые записи в любой последовательности. С помощью перемещения указателя можно редактировать записи в файле. Пусть, например, нужно изменить одно из чисел, записанных в файле, умножив его на 10. Это можно сделать, если открыть файл в режиме чтения и записи (например, "rb+"), установить позицию, соответствующую изменяемому числу, и выполнить операторы:

```
fread(&a,sizeof(double),1,F);
a *= 10;
fseek(F,-sizeof(double),1);
fwrite(&a,sizeof(double),1, F);
```

Первый из этих операторов читает число в переменную a, второй – умножает его на 10. Третий оператор возвращает текущую позицию на одну запись назад, поскольку после выполнения fread позиция сдвинулась вперед. Последний оператор пишет в ту позицию, в которой было прочитано число, новое значение.

Ту же задачу можно решить иначе:

```
long int pos = ftell(F);              //Запоминание позиции
fread(&a,sizeof(double),1,F);     //Чтение одного числа из файла в переменную a
a *= 10;
fseek(F,pos,0);                       //Восстановление  позиции
fwrite(&a,sizeof(double),1, F); //Запись числа из переменной a в файл
```

Здесь функция ftell запоминает позицию, из которой читается число, а функция fseek восстанавливает эту позицию перед записью измененного числа.

3.2 Пример проекта с бинарным файлом

Пример 3.1

Создать файл, содержащий информацию о товарах: наименование товара, цена и количество. Отобрать товары, количество которых менее 10 штук. Отредактировать одну из записей. Удалить одну из записей. Записи для редактирования и для удаления ввести с клавиатуры.

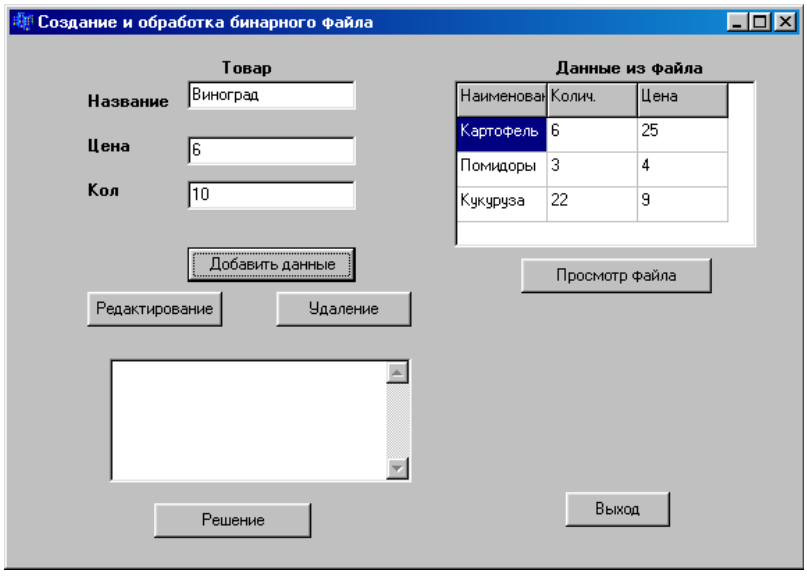


Рис. 3.1

На рис. 3.1 представлено окно формы проекта с результатами работы программы.

```
int n, flag;
 /* если добавляем данные в конец файла, то flag должен быть равен 0,
а если редактируем данные, – то flag имеет любое значение, отличное
от 0;
n будет определять номер записи в файле, причем номер будем считать
по количеству строк, отраженных в компоненте StringGrid, хотя в
файле отсчет начинается с нуля*/
FILE *f;
char s[]="a.dat";
struct tovar{
     char N[20];
     float price;
     int kol;
};
tovar t;
//---------------------------------------------------------------------------
//Кнопка «Добавить данные»
///( Добавить данные или сохранить изменения после редактирования)
void __fastcall TForm1::Button1Click(TObject *Sender)
{ if (flag==0)
{
        if ((f=fopen(s,"ab+"))==NULL)
        //либо создаем файл, либо, если он уже создан, перемещаемся
        // в конец для записи
                {ShowMessage("Файла нет!");   return;}
        strcpy(t.N,Edit1->Text.c_str( ));
        t.price=StrToFloat(Edit2->Text);
        t.kol=StrToInt(Edit3->Text);
        fwrite(&t,sizeof(tovar),1,f);
        fclose(f);
}
else
{       if ((f=fopen(s,"rb+"))==NULL)
                {ShowMessage("Файла нет!");    return;}
        strcpy(t.N,Edit1->Text.c_str( ));
        t.price=StrToFloat(Edit2->Text);
        t.kol=StrToInt(Edit3->Text);
        fseek(f,(n-1)*sizeof(tovar),0);
        //перемещаемся на запись с номером (n-1)
        fwrite (&t,sizeof(tovar),1,f);
        Button1->Caption="Добавить данные";
        fseek(f,0,0);
```

```
        int i=2;
        while (fread(&t,sizeof(tovar),1,f))
        {       SG1->RowCount=i;
                SG1->Cells[0][i-1]=AnsiString(t.N);
                SG1->Cells[1][i-1]=FloatToStr(t.price);
                SG1->Cells[2][i-1]=IntToStr(t.kol);
                i++;
        }
         fclose(f);
        flag=0;
}
Edit1->Clear( );
Edit2->Clear( );
Edit3->Clear( );
Edit1->SetFocus( );
}
//---------------------------------------------------------------------------
```

//Кнопка «Просмотр файла»

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
 if ((f=fopen(s,"rb"))==NULL)
        { ShowMessage("Файла нет!");   return; }
 SG1->Cells[0][0]="Наименование";
 SG1->Cells[1][0]="Колич.";
 SG1->Cells[2][0]="Цена";
 int i=2;
 while (fread(&t,sizeof(tovar),1,f))
 {      SG1->RowCount=i;
        SG1->Cells[0][i-1]=AnsiString(t.N);
        SG1->Cells[1][i-1]=FloatToStr(t.price);
        SG1->Cells[2][i-1]=IntToStr(t.kol);
        i++;
  }
  fclose(f);
}
//---------------------------------------------------------------------------
```

//Кнопка «Удаление» *(Удалить заданную строку в файле)*

```
void __fastcall TForm1::Button7Click(TObject *Sender)
{ FILE* tmp;                    //для работы с временным файлом
   int k;
   if ((f=fopen(s,"rb"))==NULL)
        {ShowMessage("Файла нет!");   return; }
   if ((tmp=fopen("b.dat","wb+"))==NULL)          //создаем временный файл
        {ShowMessage("Файла нет!");  return;}
   AnsiString s;
```

```
  s=InputBox("Какую запись желаете удалить?","Укажите номер строки","");
                    //здесь отсчёт номеров строк ведётся с нуля
   if (s=="") {ShowMessage("Вы не ввели данные");return;}
   k=StrToInt(s);
   int kk;
   while ((fread(&t,sizeof(tovar),1,f)))
    {
       kk=ftell(f)/sizeof(tovar);
       //ftell(f) – количество прочитанных байтов,
       //поэтому kk определяет текущий номер записи
       if (k==kk)  continue;
       //Пропускаем указанную запись; здесь отсчет ведется от единицы
       fwrite(&t,sizeof(tovar),1,tmp);
    }
    fclose(f);
    fseek(ff,0,0);                  //Переходим на начало файла
    f=fopen("a.dat","wb+");         //Вновь создаем файл для чтения и записи
                              //и копируем в него данные из вспомогательного файла
    while ((fread(&t,sizeof(tovar),1,tmp)))
       fwrite(&t,sizeof(tovar),1,f);
    fclose(tmp);
    remove("b.dat");                //Удаляем вспомогательный файл
    fseek(f,0,0);                   //Переходим на начало файла
    int i=2;
    while (fread(&t,sizeof(tovar),1,f))
    {   SG1->RowCount=i;
       SG1->Cells[0][i-1]=AnsiString(t.N);
       SG1->Cells[1][i-1]=FloatToStr(t.price);
       SG1->Cells[2][i-1]=IntToStr(t.kol);
       i++; }
     fclose(f);
}
//---------------------------------------------------------------------------
//Кнопка «Редактирование» (Редактирование выбранной записи)
void __fastcall TForm1::Button5Click(TObject *Sender)
{
if ((f=fopen(s,"rb+"))==NULL)
        {ShowMessage("Файла нет!");
        return; }
AnsiString s;
s=InputBox("Какую строку желаете корректировать?",
                        "Укажите номер строки","");
if (s=="") {ShowMessage("Вы не ввели данные");return;}
n=StrToInt(s);
fseek(f,(n-1)*sizeof(tovar),0);  //Переходим на (n-1)-ю запись
```

```
fread(&t,sizeof(tovar),1,f);
Edit1->Text=AnsiString(t.N);
Edit2->Text=FloatToStr(t.price);
Edit3->Text=IntToStr(t.kol);
fclose(f);
flag=1;
Button1->Caption="Сохранить изменения";
}
//---------------------------------------------------------------------------
//Кнопка «Решение»
//(Отобрать товары, количество которых меньше 10 штук)
void __fastcall TForm1::Button3Click(TObject *Sender)
{
if ((f=fopen(s,"rb+"))==NULL)
        {ShowMessage("Файла нет!");
        return; }
Memo1->Lines->Clear( );
Memo1->Lines->Add("Товары, количество которых меньше 10");
Memo1->Lines->Add("");
while (fread(&t,sizeof(tovar),1,f))
{       if (t.kol<10)
        Memo1->Lines->Add(AnsiString(t.N)+" "+FloatToStr(t.price)+"
                                "+IntToStr(t.kol));
}
fclose(f);
}
//---------------------------------------------------------------------------
//Кнопка «Выход»
void __fastcall TForm1::Button4Click(TObject *Sender)
{
Close( );
}
```

3.3 Файловый ввод/вывод с использованием дескрипторов

При создании программ для Windows более корректно применять стандартные функции этой системы. Для работы с ними не требуется создавать специальные переменные файлового типа: в системе Windows каждый файл имеет уникальный цифровой идентификатор (целого типа). По-английски он называется Handle и под таким названием присутствует в описании многих функций.

Файл создается с помощью функции

int **FileCreate**(const AnsiString FileName);

Например:

int f = FileCreate("a.dat");

В Windows режимы создания и открытия файла различаются.

Функция возвращает идентификатор файла (целое положительное число) или значение –1, если создать файл не удалось. Значение –1 обозначает ошибку для большинства функций Windows. Параметр ***FileName*** содержит имя файла, возможно, с полным путем поиска.

Открытие файла выполняется функцией

int **FileOpen**(const AnsiString FileName, unsigned Mode);

Режим открытия определяется параметром ***Mode***. Чаще всего применяется одно из трех следующих значений этого параметра:

fmOpenRead – открытие только для чтения из файла;

fmOpenWrite – открытие только для записи в файл;

fmOpenReadWrite – открытие и на запись и на чтение (режим изменения файла).

Запись данных в файл осуществляется функцией

int **FileWrite**(int Handle, const void *Buffer, unsigned Count);

а чтение –

int **FileRead**(int Handle, void *Buffer, unsigned Count);

Файл закрывается с помощью функции:

void **FileClose**(int Handle);

Перемещение указателя файла выполняется функцией:

int **FileSeek**(int Handle, int Offset, int Origin);

Параметр ***Offset*** задает сдвиг в байтах относительно позиции, определяемой параметром ***Origin***. Значения параметра ***Origin*** представлены в табл. 3.1.

Таблица 3.1 − Значения параметра Origin

Значение параметра Origin	Способ отсчета
0	От начала файла
1	От текущей позиции
2	От конца файла

Рассмотрим пример открытия файла и перевода указателя файла в конец файла:

```
int f;                  //дескриптор файла
f=FileOpen("a.dat", fmOpenWrite);          //Открываем файл для записи
int kb= FileSeek(f,0,2);          //Устанавливаем указатель на конец файла
                                  // и одновременно определяем размер файла
int k=25;
FileWrite(f, &k,sizeof(int));
          //Записываем в файл значение k  целого типа, размером sizeof(int)
FileClose(f);              //Закрываем файл
```

Открытие файла для чтения выполняется в следующей последовательности:

```
f=FileOpen("a.dat",fmOpenRead);
FileRead(f, указатель на тип читаемых данных, количество читаемых байтов);
```

```
FileClose(f);
```

Открытие файла для чтения и записи выполняется в следующей последовательности:

```
f=FileOpen("a.dat",fmOpenReadWrite);
if (f != -1)
/*Можно работать с файлом: читать данные, или записывать, установив указатель записи с помощью функции FileSeek на нужную позицию в файле*/
```

В заключение отметим, что с помощью рассмотренного метода создания и обработки файлов можно работать с данными любого типа.

Рассмотрим пример создания и обработки бинарного файла с помощью дескриптора.

3.4 Пример создания и обработки бинарного файла с использованием дескрипторов

Пример 3.1

Создать файл с информацией об авиарейсах: номер рейса, маршрут движения, время отправления (по шаблону ЧЧ:ММ), время прибытия в пункт назначения (по шаблону ЧЧ:ММ).

С клавиатуры вводятся: номер рейса, маршрут движения, время отправления и длительность полёта. Время прибытия в пункт назначения вычислять автоматически.

Предусмотреть возможность удаления данных о ночных рейсах (с 00:00 по 5:00).

Отсортировать данные по времени отправления самолетов.

Внешний вид окна рабочей формы при просмотре исходных данных в файле показан на рис. 3.2.

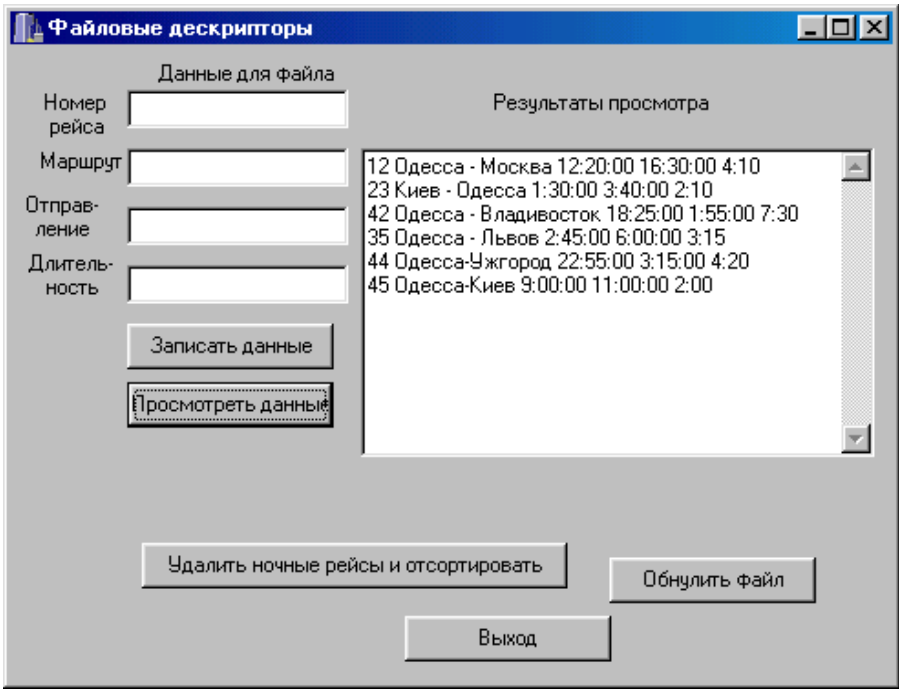


Рис. 3.2

Результаты удаления ночных рейсов и сортировка данных по одному из полей показаны на рис. 3.3.

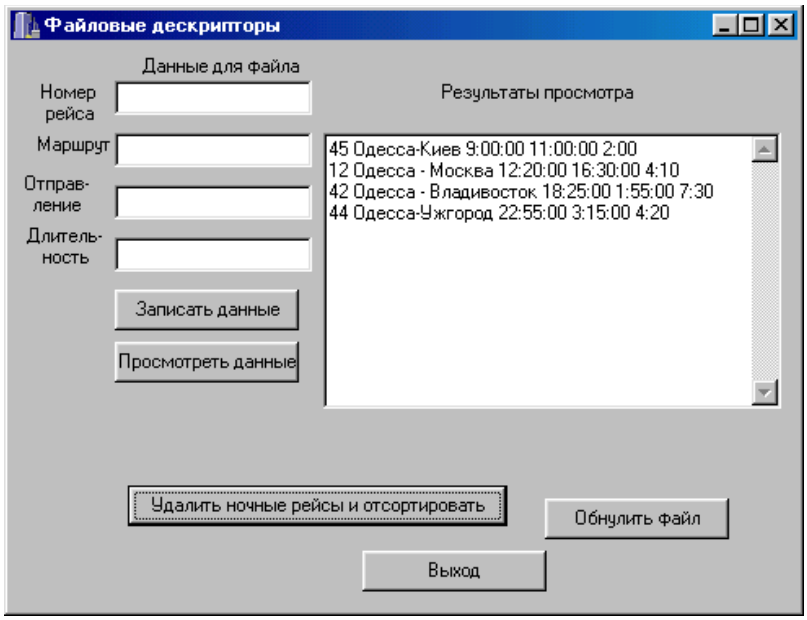


Рис. 3.3

Текст программы

```
// Глобальные объявления
struct avio_struct{
        char  num[10],
        marshrut[30],
        all_time[6];
        TDateTime start_time,
        last_time;
        AnsiString info( )
                {return AnsiString(num)+" " + (AnsiString)marshrut+" "
                + start_time.TimeString( )+" "+ last_time.TimeString( )+" "
                + (AnsiString)all_time;
                }
};
int size = sizeof(avio_struct);
avio_struct as;
int f;
char s[]="a.dat";
//---------------------------------------------------------------------------
//Конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner) : TForm(Owner)
{
 if (FileExists(s))
        f=FileOpen(s,fmOpenReadWrite);
```

```
        //либо открываем для чтения и записи уже существующий файл
 else   f=FileCreate(s);
        //либо создаем новый файл
 FileClose(f);
}
//---------------------------------------------------------------------------
//Кнопка «Записать данные»
{
AnsiString s1,s2;
int n;
strcpy(as.num,Edit1->Text.c_str( ));
strcpy(as.marshrut,Edit2->Text.c_str( ));
as.start_time=Edit3->Text;
strcpy(as.all_time,Edit4->Text.c_str( ));
s1=(AnsiString)as.all_time;
n=StrToInt(s1.SubString(1,s1.Pos(':')-1));
if (n>=24)n=n%24;
s2=IntToStr(n)+":"+s1.SubString(s1.Pos(':')+1,2);  //Создаем шаблон ЧЧ:ММ
as.last_time=as.start_time+s2;
f=FileOpen(s,fmOpenReadWrite);
if ( f != -1 )          //Проверяем корректность открытия файла
{
       //Файл открыт для чтения и записи
       FileSeek(f,0,2);              //Устанавливаем указатель на конец файла
       FileWrite(f,&as,size);        //и записываем данные
       FileClose(f);
 }
else
{
      //Ошибка доступа к файлу
       ShowMessage("Ошибка доступа к файлу");
       return;
}
Edit1->Clear( );
Edit2->Clear( );
Edit3->Clear( );
Edit4->Clear( );
Edit1->SetFocus( );
}
//---------------------------------------------------------------------------
//Кнопка «Просмотреть данные»
{
Memo1->Clear( );
f=FileOpen(s,fmOpenRead);
while (FileRead(f,&as,size))
```

```
//Открываем файл для чтения и читаем данные, пока они есть в файле
Memo1->Lines->Add(as.info( ));
FileClose(f);
}
//---------------------------------------------------------------------------

//Кнопка «Удалить ночные рейсы и отсортировать»
{
TDateTime d1,d2;
d1=TDateTime("00:00");
d2=TDateTime("05:00");
int fkol;
Memo1->Clear( );
f=FileOpen(s,fmOpenReadWrite);
fkol=FileSeek(f,0,2);      //Определяем общее количество байтов в файле
int kol=fkol/size;  //Определяем количество записей в файле
avio_struct temp;  //Вспомогательная переменная типа структуры
                   //для сортировки
avio_struct *av=new avio_struct[kol];  //Выделяем динамическую память
                                       //под массив структур
int kolzap=0;  //Определяем количество записей с учетом пропуска данных
                 //о ночных рейсах
FileSeek(f,0,0);    //Устанавливаем указатель записи на начало файла
int i=0;
while (FileRead(f,&as,size))
{       if (as.start_time>=d1 && as.start_time<=d2) continue;
//Пропускаем записи с ночными рейсами
        *(av+kolzap)=as;          //Сохраняем запись в элементе массива
        kolzap++; //Определяем истинное количество записей без ночных рейсов
}
FileClose(f);
for (i=0;i<kolzap-1;i++)
        for (int j=i+1;j<kolzap;j++)
                if (av[j].start_time<av[i].start_time)          //Сортируем данные
                {       temp=av[i];
                        av[i]=av[j];
                        av[j]=temp;
                }
f=FileCreate(s);                  //Обнуляем файл и закрываем его
FileClose(f);
f=FileOpen(s,fmOpenWrite);   //Открываем файл для записи
for (i=0;i<kolzap;i++)
        FileWrite(f,&av[i],size); //Записываем данные в обнуленный файл
for (i=0;i<kolzap;i++)            //Просматриваем, что записали в файл
        delete av;                //Освобождаем динамический массив
Button2Click(NULL); //Просматриваем  данные после удаления
//и сортировки
}
```

Лабораторная работа № 4
Бинарные файлы

Цель работы: Научиться создавать и обрабатывать бинарные файлы в C++.

Контрольные вопросы

1. В чём заключается отличие текстового файла от бинарного?
2. Как открыть бинарный файл?
3. Как выполняется чтение из бинарного файла?
4. Как выполняется запись в бинарный файл?
5. Как определить текущее положение указателя в бинарном файле?
6. Как переместиться в определённое место в памяти?
7. Приведите последовательность действий при редактировании записи в бинарном файле.
8. Приведите последовательность действий при удалении записи из бинарного файла.
9. Предложите как можно больше способов для добавления новой записи в конец существующего файла.
10. Как определить размер файла в байтах? Предложите несколько способов.

Лабораторное задание

Подготовить протокол лабораторной работы с указанием данных, составляющих задание, структурную схему и текст программы, а также ожидаемые результаты в виде таблицы.

Создать проект программы, которая:

Создаёт бинарный файл с полями, указанными в таблицах Л.4.1, Л.4.2 и Л.4.3, и заполняет его данными согласно индивидуальному варианту. Данные вводятся пользователем с клавиатуры по одной записи и добавляются в файл.

Выводит на форму созданный файл.

Выполняет с данными, записанными в файле, операции согласно индивидуальному варианту. При выполнении заданий из табл. Л.4.3 использовать файловые дескрипторы.

Индивидуальные задания среднего уровня сложности

Таблица Л.4.1 − Варианты индивидуальных заданий

№ вар	Содержание бинарного файла	Задание
1	Список сотрудников предприятия: табельный номер, ФИО, должность, дата приёма на работу, оклад	Вывести данные о техниках, которые проработали более 5-ти лет, и внести корректировку их оклада, увеличив его на 20 %

Продолжение таблицы Л.4.1

№ вар	Содержание бинарного файла	Задание
2	Сведения об игроках команды «Черноморец»: номер игрока, ФИО, амплуа (вратарь, защитник и т. д.), количество забитых голов за сезон	Вывести данные о нападающих, которые за сезон не забили ни одного гола, и исключить их из команды. Добавить нового перспективного игрока
3	Список абонентов телефонной сети: текущий номер абонента, ФИО, адрес, номер телефона (вводить по шаблону ###-##-##), задолженность по оплате, количество дней задолженности	Добавить в файл абонентов новой АТС, номера которых начинаются с 734. Вывести данные об абонентах, у которых есть задолженность более 20 дней
4	Список предприятий города, выпускающих электронное оборудование: наименование завода, вид продукции, количество выпущенной продукции за квартал, рентабельность предприятия (если предприятие убыточное, то значение рентабельности отрицательное)	Составить список предприятий, выпускающих наибольшее количество телевизоров. Удалить данные о нерентабельных предприятиях
5	Список нереализованного товара на складе: наименование товара, количество, цена за единицу товара, закупщик товара, сумма долга за товар	Удалить из файла товары, по которым нет задолженности, и добавить в файл нереализованные товары, которые снова появились на складе
6	Список сведений о транзисторах: порядковый номер, тип транзистора, материал, коэффициент передачи тока, минимальное и максимальное значения	Вывести данные о транзисторах, у которых коэффициент передачи в пределах 50..70 ед. Добавить данные о германиевых транзисторах
7	Список студентов в группе: номер по журналу, ФИО, рейтинг по математике, физике, информатике, средний балл по предметам по 100-балльной системе (вычислять при вводе данных)	Удалить исключенных из группы студентов, у которых средний балл ниже 20. Добавить в список двух новых студентов
8	Составить список научных работников научно-исследовательского центра: ФИО, должность (по шаблону: инженер, мнс, снс, внс), научное звание, количество научных работ	Вывести список работников и перевести мнс в снс, у которых более 10-ти научных работ, а инженеров − в мнс, у которых более 5-ти научных работ

Продолжение таблицы Л.4.1

№ вар	Содержание бинарного файла	Задание
9	Список электронных компонентов продукции: номер, наименование, для какой продукции используется, в каком количестве, цена	Вывести данные о количестве конденсаторов, которые используются в телевизорах и радиоприемниках, и определить их общую стоимость
10	Список средств индивидуальной защиты от поражения электрическим током: инвентарный номер, наименование, год испытания (по шаблону: ДД-ММ-ГГ), каким напряжением испытано	Вывести данные о средствах защиты, которым до испытания остался один месяц, и добавить в файл данные о новых средствах защиты
11	Сведения о высокочастотных кабелях: номер, тип кабеля, погонная ёмкость, погонное сопротивление, рабочее напряжение, внешний диаметр	Вывести все данные о кабелях, тип которых начинается с букв «РК». Удалить сведения о кабелях, у которых рабочее напряжение меньше 100 В
12	Список используемых ЭВМ в компьютерном классе: номер по порядку, тип ЭВМ, тактовая частота, оперативная память, ёмкость диска, год установки в классе	Вывести данные об ЭВМ типа Celeron с тактовой частотой 2000 МГц. Удалить данные об ЭВМ типа Pentium-1
13	Список музыкальных дисков в звукостудии: порядковый номер, название диска, количество композиций, длительность звучания, цена диска	Вывести данные о дисках, длительность звучания которых от 40 до 50 мин, и добавить сведения о новых дисках
14	Список сотрудников предприятия: табельный номер, ФИО, должность, год рождения, пол (по шаблону: муж – мужчины, жен – женщины), адрес	Вывести данные о женщинах, которым остался один год до пенсии, и уволить мужчин, возраст которых превышает 75 лет
7	Список книг в книгохранилище: инвентарный номер, название, год издания, цена	Вывести сведения о книгах по программированию, в названии которых упоминается язык С++
16	Сведения об автомобилях для продажи: номер по списку, тип автомобиля, тип двигателя, километраж пробега, стартовая цена продажи, год выпуска	Вывести данные об автомобилях, у которых пробег меньше 50 000 км и дата выпуска меньше двух лет давности. Удалить данные об автомобилях типа «седан» с дизельным двигателем

Продолжение таблицы Л.4.1

№ вар	Содержание бинарного файла	Задание
17	Список фильмов в фильмотеке: порядковый номер, название фильма, продолжительность демонстрации, год выпуска, количество копий	Вывести данные о фильмах, срок хранения которых на складе в текущем году составляет 5 лет, и удалить сведения о фильмах, которые хранятся более 10-ти лет
18	Сведения о сезонном распределении температуры воздуха в городе: дата (по шаблону: ДД-ММ-ГГ), средняя температура на протяжении суток, максимальное отклонение температуры от средней	Вывести данные о днях, когда максимальное отклонение превышает 10 градусов, и добавить сведения о температуре за прошедшие сутки
19	Сведения об абонентах сотовой сети: порядковый номер, ФИО, номер телефона, тип телефона, дата оплаты за телефон	Вывести данные об абонентах, которые не платят за телефон последние два месяца, и удалить данные о них из файла
20	Сведения о статистической ошибке канала телеизмерений (ТИ) потерь электроэнергии: порядковый номер, дата (по шаблону ДД-ММ-ГГ), наименование направления линии, показания электросчетчика, показания канала ТИ	Вывести данные об абсолютной ошибке за предыдущий день и внести данные за новый отчетный день
21	Сведения о медикаментах в аптеке: порядковый номер, наименование лекарства, срок годности (по шаблону ДД-ММ-ГГ), цена	Вывести данные о лекарствах, срок годности которых истекает в текущем году, и вычислить сумму предполагаемых убытков. Удалить из файла сведения о просроченных лекарствах
22	Сведения об отказах линии связи: направление каналов связи, дата отказа (по шаблону: ДД-ММ-ГГ), время отказа: начало-конец (по шаблону: ЧЧ-ММ-СС)	Вывести данные обо всех отказах, длительность которых превышает 10 мин. Добавить данные об отказах на направлениях линий связи Одесса–Киев
23	Статистические данные о работе каналов ионосферной связи: порядковый номер, направление связи, месяц, часы наилучшей работы на этом направлении (по шаблону: ЧЧ-ММ)	Вывести данные об общей длительности наилучшей работы за февраль. Удалить данные по неперспективному направлению Одесса–Красноярск

Окончание таблицы Л.4.1

№ вар	Содержание бинарного файла	Задание
24	Сведения об услугах почтовой связи: порядковый номер, вид отправлений, размер оплаты за пересылку наземным транспортом, размер оплаты за пересылку воздушным транспортом, время доставки	Вывести данные о видах отправлений, для которых время доставки не превышает пяти суток. Добавить данные об отправлениях денежных переводов
25	Сведения о движении поездов: номер поезда, направление, тип поезда, время отправления (по шаблону: ЧЧ-ММ), цена билета	Вывести данные обо всех поездах, которые отправляются с 10:00 до 17:00. Выяснить, есть ли данные о направлении Одесса–Киев
26	Сведения об очереди на установку телефона: порядковый номер, ФИО, номер паспорта, дата подачи заявления, наличие льгот	Вывести данные о людях, которые подали заявление в 1998 году, и добавить данные о тех, кто имеет льготы
27	Сведения об оплате коммунальных услуг: порядковый номер, ФИО, адрес, дата оплаты, наличие долга	Вывести данные о тех, кто своевременно вносит плату, и удалить данные о должниках, которые оплатили долг
28	Сведения об аварийных домах в городе: порядковый номер, улица, номер дома, год возведения, год последнего капитального ремонта	Вывести данные о домах, которые были построены до 1927 года. Добавить данные о домах, в которых за последние 10 лет не проводился капитальный ремонт
29	Данные Госавтоинспекции об украденных автомобилях: порядковый номер, марка автомобиля, государственный номер, цвет, дата кражи	Вывести данные об автомобилях, украденных за последний год, и удалить данные об автомобилях, которые были найдены
30	Список принтеров для продажи: порядковый номер, тип принтера, фирма-изготовитель, скорость печати, качество печати, средняя цена в текущем году	Вывести сведения о высококачественных принтерах. Добавить данные о наиболее скоростных принтерах

Индивидуальные задания повышенного уровня сложности

Таблица Л.4.2 − Варианты индивидуальных заданий

№ вар	Содержание бинарного файла	Задание
1	Список работников предприятия: табельный номер, ФИО, пол, дата рождения, должность, дата приема на работу, оклад	Вывести сведения о сотрудниках, до дня рождения которых осталось 10 дней
2		Вывести сведения о сотрудниках, день рождения которых отмечается в текущем месяце
3		Найти самого молодого по возрасту сотрудника
4		Вывести сведения о сотрудниках, которые в текущем году празднуют юбилей, т. е. им исполняется количество лет, кратное 5-ти
5		Вывести сведения о сотрудниках-пенсионерах, т. е. женщинах, которым уже исполнилось 55 лет, и мужчинах, которым исполнилось 60 лет
6		Вывести сведения о сотрудниках, день рождения которых отмечается в марте
7		Вывести сведения о сотрудниках, стаж работы которых превышает 5 календарных лет (1 год – ≈365 дней)
8		Вывести сведения о сотрудниках, которые поступили на работу в возрасте до 20 лет
9		Вывести сведения о сотрудниках, которые в текущем году празднуют юбилейный стаж работы, т. е. проработали кратное 5-ти количество лет
10		Найти сотрудника с наибольшим стажем работы
11	Список товаров в магазине: номер по порядку, наименование, дата изготовления, изготовитель, дата конечной реализации	Вывести сведения о сроке годности (длительность в днях) всех товаров
12		Вывести сведения о товарах, срок годности которых истекает в ближайшие 4 дня
13		Вывести сведения о товарах, произведенных в прошлом месяце
14		Вывести сведения о товарах, для которых прошла половина срока годности
15		Найти товар с максимальным сроком годности (длительность в днях)
16		Вывести сведения о скоропортящихся товарах, срок годности которых до 5-ти дней
17		Вывести сведения о просроченных товарах, срок годности которых уже истек

Окончание таблицы Л.4.2

№ вар	Содержание бинарного файла	Задание
18	Расписание поездов: номер поезда, направление, тип поезда, дата и время отправления, дата и время прибытия на конечную станцию	Вывести сведения о поездах, которые отправляются в промежутке от 17:00 до 20:00
19		Найти поезд по направлению «Одесса–Киев» с минимальным временем пребывания в пути
20		Вывести сведения о поездах, которые находятся в пути дольше суток
21		Вывести сведения о поездах по направлению «Одесса-Киев», которые прибывают в Киев с 6:00 до 8:00
22		Вывести сведения о поездах, которые отправляются из Одессы, и вычислить время их пребывания в пути
23		Найти поезд с максимальным временем пребывания в пути
24		Найти поезд по направлению «Киев–Одесса», который позже других приходит в Одессу
25	Сведения об оплате коммунальных услуг: ФИО, адрес, дата последней уплаты за услуги, долг	Вывести сведения об абонентах, которые не платили уже более трёх месяцев, и вычислить суммарный долг
26		Вывести сведения об абонентах, которые в текущем месяце произвели оплату, но имеют долг
27		Вывести сведения об абонентах, которые произвели оплату позднее 10-го числа текущего месяца, и начислить каждому из них пеню в размере 0,1 % от долга за каждый просроченный день
28		Вывести сведения об абонентах, которые произвели оплату до 10-го числа текущего месяца вперёд (значение долга отрицательное)
29	Сведения об оплате коммунальных услуг: ФИО, адрес, дата последней уплаты за услуги, долг	Вывести сведения об абонентах, которые не производили оплату в течение последних 30-ти дней и имеют долг более 1 000 грн.
30		Вывести сведения об абоненте, который произвёл оплату позже других

Индивидуальные задания высокого уровня сложности

Таблица Л.4.3 − Варианты индивидуальных заданий

№ вар	Содержание бинарного файла	Задание
1	Список сотрудников предприятия: табельный номер, ФИО, должность, год рождения по шаблону ДД.ММ.ГГГГ, пол сотрудника по шаблону: муж., жен.	Создать новый файл, содержащий список сотрудников, которые в текущем году выходят на пенсию. Удалить из списка (уволить) сотрудников мужского пола, которым исполнилось 70 лет, и женщин – достигших 65-летнего возраста
2	Список лекарственных ампул: название, срок действия после вскрытия по шаблону ЧЧ:ММ, срок хранения (годы) нераспакованной ампулы	Создать второй файл, содержащий информацию о вскрытых ампулах: название лекарства, время вскрытия (вводятся пользователем) и срок, до которого ампула должна быть использована (рассчитывается согласно данным из первого файла). Удалить из второго файла информацию об ампулах, срок хранения которых в открытом виде закончилось
3	Список бытовой техники в офисе: наименование, дата покупки, срок гарантии в днях	Определить, находится ли ещё на гарантии введённое оборудование. Перенести в другой файл информацию о технике, у которой закончился срок гарантии
4	Сведения о расписании занятий в учебном заведении в течении дня: название предмета, время начало пары по шаблону: ЧЧ:ММ, время окончания пары	Длительность пары составляет 1 ч 45 мин, а перерыв между парами не более 45-ти и не менее 5-ти минут. При вводе данных выполнять проверку начала каждой пары (не накладывается ли на предыдущую пару, соблюдены ли ограничения на длительность перерыва), а конец пары рассчитывается автоматически
5	Список абитуриентов: фамилия, дата рождения и пол	Удалить из списка абитуриентов, которые старше 35 лет. Создать новый файл, в который занести информацию о юношах призывного возраста (от 18 до 27 лет)
6	Сведения о расписании движения пригородных поездов по нескольким направлениям (по каждому направлению по 3-5 рейсов в течение дня): номер рейса, направление движения, время отправления, время прибытия в конечный пункт	Из составленного расписания сформировать зимнее расписание (новый файл), в котором должны быть только утренние (до 10:00) и вечерние (после 18:00) рейсы

Продолжение таблицы Л.4.2

№ вар	Содержание бинарного файла	Задание
7	Сведения о телефонных переговорах: номер телефона, начало и конец переговоров по шаблону: ЧЧ:ММ	Из исходного файла создать два новых, в одном из которых должны быть зафиксированы дневные переговоры (с 6:00 до 20:00), а в другом – ночные
8	Список товаров в магазине: наименование, дата выпуска, конечный срок годности	Создать два новых файла: в один из которых поместить скоропортящиеся товары (срок хранения которых не более 5-ти суток), во второй – товары длительного хранения. Удалить товары, срок годности которых закончился
9	Список лекарственных ампул: название, срок действия после вскрытия по шаблону ЧЧ:ММ, срок хранения нераспакованной ампулы в годах	Удалить информацию об ампулах, срок хранения которых превысил один год, и выдать предупреждения об окончании срока хранения ампулы за 10 дней
10	Список клиентов парикмахерской на день: имя клиента, время в формате ЧЧ:ММ и предполагаемая длительность процедуры	При создании файла проверять, не занято ли запрашиваемое время и достаточно ли у мастера свободного времени для выполнения требуемой процедуры. Вывести список всех клиентов, которые придут после 16:30
11	Список пациентов на приём к врачу: фамилия, дата предыдущего посещения и время, на которое пациент записан	Удалить из файла записи о пациентах, время приёма которых уже прошло. Создать два новых файла: в один занести сведения о вторичных больных (предыдущее посещение в течение 10-ти последних дней), а во второй – об остальных
12	Список дел на текущий день: условное название, время начала, предполагаемая длительность	Занести в новый файл информацию о делах, которые нужно выполнить с 12:45 до 17:30. Вывести информацию о свободном времени (начало и конец временного промежутка и рассчитать длительность)
13	Список телефонных переговоров: время начала звонка, время окончания разговора	Рассчитать оплату за переговоры, считая, что минута разговора в дневное время (с 9:00 до 20:00) стоит 1,5 грн., а в ночное – 0,90 грн. Удалить из файла данные о разговорах длительностью менее 3 мин

Продолжение таблицы Л.4.2

№ вар	Содержание бинарного файла	Задание
14	Расписание движения междугородных автобусов: пункт назначения, время отправления и длительность пути (в часах и минутах)	Удалить из файла информацию о рейсах, у которых хотя бы часть пути попадает на ночное время (с 23:00 до 6:00). Исключение составляют маршруты с длительностью пути более 17 часов. Определить время отправления последнего автобуса в заданный пункт назначения
15	Список сотрудников предприятия: фамилия, дата рождения, дата приёма на работу	Вывести список сотрудников, которые старше 40 лет, и сотрудников, которые работают на предприятии не менее 20 лет. Удалить из файла информацию о сотрудниках, которые работают менее года
16	Список сотрудников предприятия: табельный номер, ФИО, должность, год рождения (по шаблону: ДД.ММ.ГГГГ), пол сотрудника (по шаблону: муж., жен.)	При добавлении данных выполнять проверку на возраст: он должен быть не менее 20-ти и не более 55-ти лет. Создать два новых файла: первый содержит список сотрудников, которым не более 40 лет, второй – список остальных
17	Список товаров в магазине: наименование, дата выпуска, конечный срок годности, цена	Снизить цену на 5 % на все товары, максимальный срок хранения которых не более двух недель. Повысить цену на 3 % на товары, максимальный срок хранения которых более одного года. Максимальный срок хранения рассчитать как разность конечного срока годности и даты выпуска. Скопировать в новый файл данные о товарах, произведенных ранее текущего месяца
18	Перечень технических перерывов в работе кассы: время начала и время конца перерыва	При вводе данных проверять, не накладывается ли новый перерыв на уже имеющийся. Определить, успеет ли кассир обслужить N (ввести с клавиатуры) человек, стоящих в очереди, если на одного клиента в среднем тратится 15 мин
19	Список произведений автора: название, дата написания, год издания (если произведение не издано, то 0)	Скопировать в отдельный файл произведения, написанные в течение последних четырёх лет. Вывести произведения, которые были напечатаны более чем через 5 лет после написания
20	Список телепередач: название, время начала и время окончания передачи	Определить длительность каждой передачи. Создать новый файл, содержащий передачи, которые показываются в дневное время (с 9:00 по 18:00)

Продолжение таблицы Л.4.3

№ вар	Содержание бинарного файла	Задание
21	Список автомобилей автосалона: название, дата выпуска, дата поступления в продажу	Создать список новых автомобилей (которые поступили в продажу не более чем через 2 месяца после выпуска). Вывести информацию об автомобилях, которые были выпущены не ранее введённого года
22	Список дел на текущий день: условное название, время начала, предполагаемая длительность	Определить, какое дело по списку следующее (ближайшее от текущего времени). Создать файл с информацией о свободном времени во второй половине дня (после 13:00): начало и окончание временного промежутка и длительность (рассчитать)
23	Список клиентов, обслуженных менеджером в течение дня: фамилия, время прихода и время ухода клиента	При вводе данных проверять их допустимость (не пересекаются ли они с имеющимися). Создать новый файл, содержащий информацию о клиентах, с которыми менеджер общался более 30 мин
24	Список товаров в магазине: наименование, дата выпуска, конечный срок годности, цена	Перенести в другой файл данные о товарах, у которых истекает срок годности (до его окончания осталось менее 10 % допустимого времени хранения). Вывести информацию о товарах, произведённых за последние 10 дней
25	Список сотрудников предприятия: фамилия, дата рождения, дата приёма на работу	Вывести список сотрудников, у которых день рождения в текущем месяце и которые проработали на предприятии не менее 5-ти лет. Создать новый файл с информацией о сотрудниках, поступивших на работу на данное предприятие в возрасте не старше 25-ти лет и проработавших на нём не менее 10-ти лет
26	Сведения о телефонных разговорах: дата, время начала и время окончания разговора	Определить размер оплаты с учётом того, что в ночное время (с 22:00 по 7:00) звонки бесплатные, в бизнес-время (с 9:00 по 18:00) стоимость минуты разговора 1,7 грн., а в остальное время – на 30 % дешевле, чем в бизнес-время
27	Список клиентов, обслуженных менеджером в течение дня: фамилия клиента, время прихода и время ухода клиента	При вводе данных проверять их допустимость (не пересекаются ли они с уже имеющимися). Определить, было ли у менеджера в течение рабочего дня (с 9:00 до 17:00) свободное время, и если да, то сколько и в какой половине дня

Окончание таблицы Л.4.3

№ вар	Содержание бинарного файла	Задание
28	Список автомобилей автосалона: название, дата выпуска, дата поступления в продажу	Создать список автомобилей, которые поступили в продажу за последний месяц. Вывести информацию о подержанных автомобилях (которые были выпущены более чем за год до поступления в продажу)
29	Список произведений автора: название, дата написания, год издания (если произведение не издано, то 0)	Определить количество «зимних» произведений автора. Перенести в отдельный файл информацию о произведениях, написанных и изданных в прошлом веке
30	Список покупателей, которые приобрели товары со скидкой в день акции: фамилия, пол, дата рождения, количество единиц товара	Предполагается, что стоимость одной единицы товара 100 грн., скидка на товар равна возрасту человека. Пенсионерам (мужчины с 60-ти лет, женщины с 55-ти лет) предоставляется дополнительная скидка 5 %. Определить выручку магазина за день. Создать новый файл с информацией о покупателях, совершивших покупку более чем на 250 грн.

4 Классы и объекты библиотеки визуальных компонентов

Процесс создания компонента может быть представлен как последовательность следующих шагов:

1 выбор компонента - родителя;
2 создание модуля компонента - потомка;
3 тестирование компонента;
4 добавление нового компонента в пакет компонентов.

Примечание: в данной лабораторной работе пункт 4 (добавление компонента) рассматриваться не будет.

4.1 Выбор компонента - родителя

Следует чётко сформулировать назначение компонента, после чего определить, какой из стандартных компонентов наиболее близок по своему назначению, виду и функциональным возможностям к компоненту, который разрабатывается.

4.2 Создание модуля компонента

Чтобы начать работу над новым компонентом, следует активизировать процесс создания нового приложения (File – New – Application), в меню Component выбрать New Component и в поле диалогового окна New Component ввести информацию о создаваемом компоненте.

Пример 4.1

Создать новый компонент – потомок класса TButton, который подсчитывает количество нажатий на кнопку Button1 во время выполнения C++Builder-проекта.

После выбора New Component, как описано выше, появится диалоговое окно New Component, представленное на рис. 4.1.

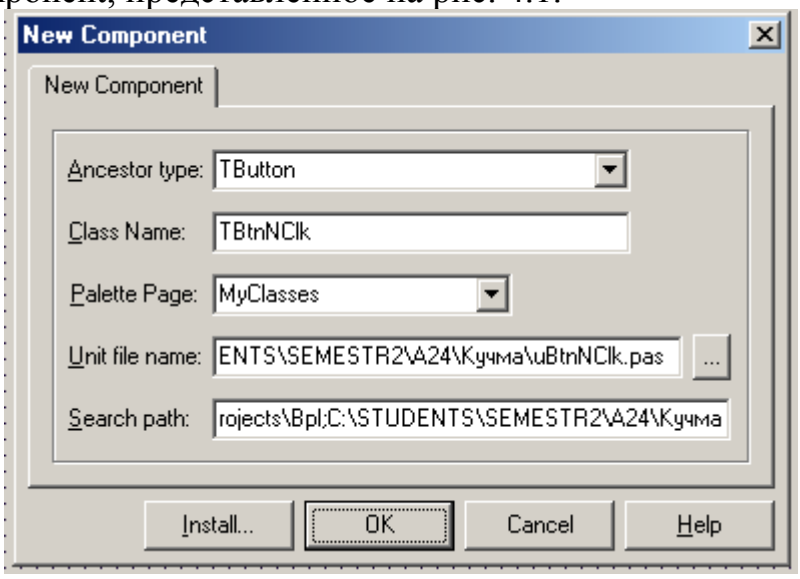


Рис. 4.1

В это диалоговое окно следует ввести информацию о создаваемом компоненте: Ancestor type, Class Name, Palette Page, Unit file name и Search path.

Ancestor type – класс, который является родительским для потомка, который строится (в данном примере выбран TButton).

Class Name – имя нового класса - потомка (например, можно написать TBtnNClk).

Palette Page – страница палитры компонент, на которую вы хотите поместить новый компонент. По умолчанию – это страница Samples. Можно указать одну из существующих страниц или же задать новое имя – соответствующая новая страница палитры будет сформирована (на рис. 4.1 новой странице палитры компонентов присваивается имя MyClasses).

Unit file name – как увидим далее, C++-текст, необходимый для построения нового класса, помещается в новый модуль (unit). Следует указать имя этого модуля (полный путь к нему). По умолчанию имя файлов модуля совпадает с именем класса, за исключением первой буквы T, следовательно, если класс называется TBtnNClk, файлы будут названы BtnNClk.cpp и BtnNClk.h (на рис. 4.1 имя BtnNClk изменено на uBtnNClk, чтобы подчеркнуть, что речь идет о unit). Что до пути к файлу, то по умолчанию C++Builder устанавливает путь к своему библиотечному каталогу LIB. Чтобы не “засорять” каталог LIB, рекомендуется изменить этот путь и разместить unit нового класса в собственном каталоге, например:

C:\STUDENTS\SEMESTR2\A24\<фамилия>\uBtnNClk.cpp

Search path – здесь перечислены каталоги, в которых C++Builder будет искать unit нового класса. Если вы указали в предыдущем пункте полный путь к вашему собственному каталогу, C++Builder автоматически допишет его к Search path; так что вам остается лишь проверить, правильно ли это было сделано.

После щелчка на кнопке ОК будет сформирован модуль нового компонента, состоящий из двух файлов: файла заголовка **uBtnNClk.h** и файла реализации **uBtnNClk.cpp**. Шаблон (начальное содержимое) файла **uBtnNClk.h**:

```
//---------------------------------------
#ifndef uBtnNClk
#define uBtnNClk
//---------------------------------------
#include <SysUtils.hpp>
#include <Controls.hpp>
#include <Classes.hpp>
#include <Forms.hpp>
#include <StdCtrls.hpp>
//---------------------------------------
class PACKAGE TBtnNClk : public TButton
{
private:
protected:
public:
      __fastcall TBtnNClk(TComponent* Owner);
__published:
```

```
};
//---------------------------------------
#endif
```

В сформированный шаблон компонента нужно внести дополнения: объявить новые поля данных, функции доступа к полям данных, свойства и методы. Если на некоторые события компонент должен реагировать не так, как его родитель, то в объявление класса нужно поместить объявления соответствующих функций обработки событий.

Окончательный текст библиотеки, которая описывает наш новый компонент, будет следующим.

Файл uBtnNClk.h:

```
#ifndef uBtnNClk
#define uBtnNClk
#include <SysUtils.hpp>
#include <Controls.hpp>
#include <Classes.hpp>
#include <Forms.hpp>
#include <StdCtrls.hpp>
//---------------------------------------
class PACKAGE TBtnNClk : public TButton
{
private:
      int NClicks;
protected:
public:
       __fastcall TBtnNClk(TComponent* Owner);
       DYNAMIC void __fastcall Click(void);
__published:
};
//---------------------------------------
#endif
```

Файл uBtnNClk.cpp:

```
#include <vcl.h>
#pragma hdrstop

#include "uBtnNClk.h"
#pragma package(smart_init)
//---------------------------------------
static inline void ValidCtrCheck(TBtnNClk *)
{
      new TBtnNClk(NULL);
}
//---------------------------------------
```

```
__fastcall TBtnNClk::TBtnNClk(TComponent* Owner)
        : TButton(Owner)
{NClicks=0;}
//---------------------------------------
void __fastcall TBtnNClk::Click(void)
{
NClicks++;
Caption=Name+" ("+IntToStr(NClicks)+")";
TButton::Click( );
}
//---------------------------------------
namespace Ubtnnclk
{
  void __fastcall PACKAGE Register( )
     {
     TComponentClass classes[1]=
                    {__classid(TBtnNClk)};
     RegisterComponents("MyClasses", classes, 0);
     }
}
```

4.3 Тестирование компонента

Перед тем как добавить новый компонент на палитру компонентов, его необходимо всесторонне проверить и убедиться, что компонент работает как следует.

Для проверки (до размещения на палитре компонентов) вновь созданный компонент может быть установлен на форме динамически, т. е. во время работы приложения. Для этого следует разместить на рабочей форме необходимые кнопки, окна и т.п., позволяющие управлять проектом в целом, после чего:

1 В текст файла реализации (.cpp-файл) включить библиотеку с текстом реализации компонента, т. е.:

```
#include "uBtnNClk.h"
```

2 Объявить компонент:

```
TBtnNClk * BtnNClk1
```

Обратите внимание: в этом объявлении BtnNClk1 будет ссылкой на объект типа TBtnNClk, поэтому объявление не обеспечивает создания экземпляра компонента, а только создаёт указатель на экземпляр.

3 Для формы Form1 создать процедуру обработки события OnCreate. В ней вызвать конструктор тестируемого компонента TBtnNClk, который создаст его экземпляр BtnNClk1 и выполнит его настройку.

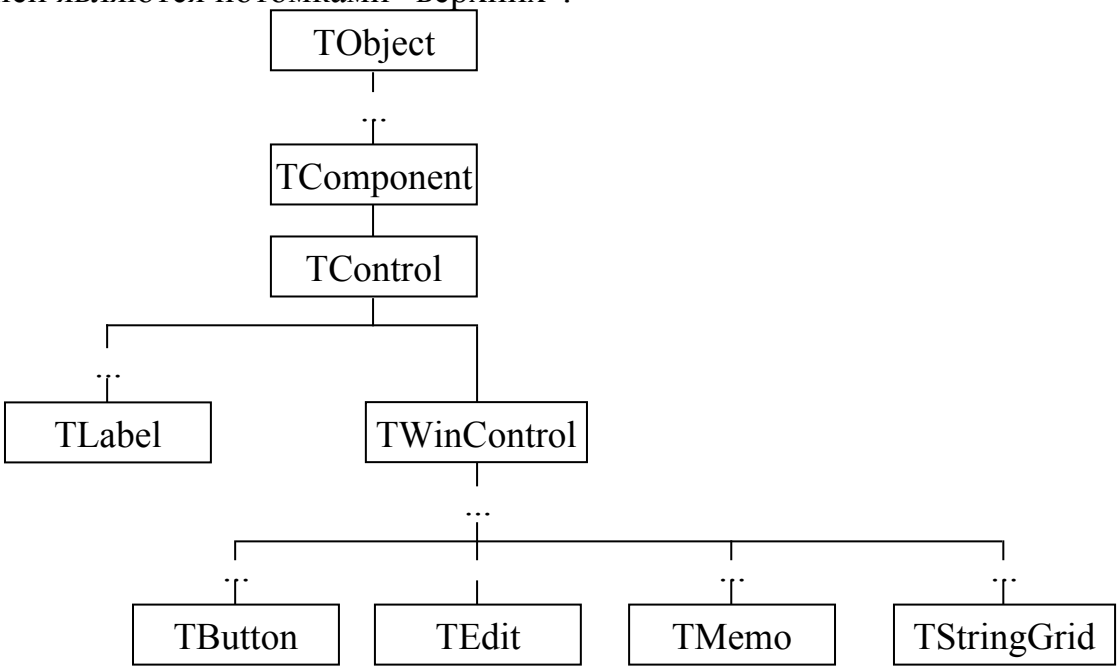
```
#include "uBtnNClk.h"
. . .
TForm1 *Form1;  //форма
TBtnNClk * BtnNClk1;  // TBtnNClk – класс, BtnNClk1 – экземпляр класса
```

```
//Обработчик события OnCreate для Form1:
void __fastcall TForm1::FormCreate(TObject *Sender)
//Создаём и инициируем компонент
{
BtnNClk1 = new TBtnNClk(Form1);
BtnNClk1->Parent=Form1;
BtnNClk1->Left=20;
BtnNClk1->Top=50;
BtnNClk1->Caption=IntToStr(NClicks);
}
```

Владельцем тестируемого экземпляра (владелец указан как параметр конструктора TBtnNClk) является окно формы Form1. Свойство Parent созданного экземпляра (его нужно обязательно задать) также равно Form1 – это означает, что новая кнопка BtnNClk1 будет размещена в окне Form1.

4.4 Иерархия классов библиотеки визуальных компонентов

Приведем фрагмент дерева классов библиотеки визуальных компонентов (VCL – Visual Components Library) системы C++Builder. Классы “нижних” уровней являются потомками “верхних”.



Класс TObject является первым в иерархии классов, принятой в библиотеке VCL, от TObject происходят все другие классы VCL. Класс TComponent инкапсулирует методы, необходимые для произвольных компонентов C++Builder. Класс TWinControl воплощает общие черты всех визуальных объектов, которыми можно управлять не только через мышь, но и с помощью клавиатуры.

В табл. 4.1 приведены заголовки некоторых методов, присущих объектам класса TControl (и, следовательно, любым потомкам этого класса). Все эти методы в классе TControl помещены в секцию protected.

Большей частью приведенные методы не вызываются прямо (т.е. через явные обращения в программе пользователя). Иногда их прямые вызовы даже

запрещены, как для метода Resize. Тем не менее все они доступны для перекрытия. Например, можно перекрыть метод Resize, заставив объект выдавать звуковой сигнал, если его размеры изменяются:

```
//В .h-файле заголовков в объявлении класса потомка TControl:
DYNAMIC void __fastcall Resize(void);
//в .cpp-файле этого потомка:
void __fastcall имя класса-потомка::Resize(void)
{
TControl:: Resize( );
Beep(300,50);          //функция, которая выдает звуковой сигнал
}                      // 300 Гц  на протяжении 50 мс
```

Таблица 4.1 – Заголовки некоторых методов объектов класса TControl

DYNAMIC void __fastcall Click(void);	Генерирует сообщение о щелчке левой кнопкой мыши
DYNAMIC void __fastcall DblClick(void);	Сообщение о двойном щелчке левой кнопкой мыши
DYNAMIC void __fastcall MouseDown(TMouseButton Button, Classes::TShiftState Shift, int X, int Y);	Сообщение о нажатии кнопки мыши. Параметр Button определяет нажатие: левой, средней или правой кнопки мыши. Shift показывает, была ли при этом нажата какая-нибудь из кнопок клавиатуры: Shift, Ctrl, Alt. X, Y – координаты мыши
DYNAMIC void __fastcall MouseMove(Classes::TShiftState Shift, int X, int Y);	Сообщение о том, что мышь движется. Параметры аналогичны приведенным выше
DYNAMIC void __fastcall MouseUp(TMouseButton Button, Classes::TShiftState Shift, int X, int Y);	Сообщение об отпускании кнопки мыши. Параметры аналогичны параметрам метода MouseDown
DYNAMIC void __fastcall Resize(void);	Сообщение о том, что размеры объекта изменились

В табл. 4.2 приведены заголовки некоторых методов класса TWinControl (обратите внимание на директиву virtual в объявлении функции SetFocus − роль этой директивы подобна роли DYNAMIC):

Таблица 4.2 − Заголовки некоторых методов класса TWinControl

virtual void __fastcall SetFocus(void);	Передает объекту “фокус”, после чего нажатия клавиш клавиатуры будут восприниматься именно этим объектом
DYNAMIC void__fastcall KeyDown(Word &Key, Classes::TShiftState Shift);	Генерирует сообщение о нажатии клавиши Key. Параметр Shift такой же, как в MouseDown
DYNAMIC void__fastcall KeyUp(Word &Key, Classes::TShiftState Shift);	Сообщение об отпускании клавиши Key. Параметр Shift такой же, как в MouseDown и KeyDown
DYNAMIC void__fastcall KeyPress(char &Key);	Сообщение о том, что клавиша Key была нажата и отпущена. В отличие от методов KeyDown и KeyUp, здесь Key – не код (номер) клавиши, а символ, который ей соответствует. Клавиши, с которыми не связаны символы (такие как Enter или F1), не обрабатываются методом KeyPress

Лабораторная работа № 5

Классы и объекты библиотеки визуальных компонентов

Цель работы: Ознакомление с понятиями класса, объекта, метода, инкапсуляции, наследования. Построение классов - потомков визуальных компонентов C++Builder.

Контрольные вопросы

1 В чём сходны и чем отличаются в C++ понятия структуры и класса?

2 Как соотносятся между собой понятия объекта и класса?

3 Как соотносятся между собой понятия объекта и экземпляра класса?

4 Приведите конкретный пример какого-нибудь класса и объектов этого класса.

5 Поля двух объектов одного и того же класса тождественны. Тождественны ли эти объекты?

6 В каком смысле можно утверждать, что методы двух объектов одного и того же класса тождественны? А в каком смысле они разные?

7 Что такое инкапсуляция? Приведите примеры.

8 Как соотносятся между собой понятия класса и компонента?

9 В чём сходны и чем отличаются методы Click и Button1Click?

10 Приведите пример задачи, которую целесообразно решать путем построения потомков класса. Объясните, почему эту задачу нельзя или нецелесообразно решать другими способами.

11 Объясните, почему имена классов начинаются, как правило, с буквы T. Можно ли начинать их с других букв?

12 Что такое Palette Page?

13 Для чего нужна секция объявления класса private? Приведите примеры полей или методов, которые целесообразно помещать в эту секцию.

14 В чём сходны и чем отличаются секции public и __published?

15 Где следует размещать прототип метода?

16 Как вызвать “родительский” метод?

17 Перечислите действия, которые следует выполнить для перекрытия метода.

18 Объясните, что такое constructor.

19 В объявлении некоторого класса слово private было использовано дважды. Для чего?

20 Какой смысл имеют параметры функции RegisterComponents?

21 Что такое C++Builder-пакет?

22 Какие действия можно выполнить в диалоговом окне C++Builder-пакета?

23 Приведите примеры иерархических зависимостей для C++Builder-классов.

24 Как можно охарактеризовать класс TControl?

25 Какая черта класса TObject отличает его от всех других классов?

26 В чём сходны и чем отличаются классы TControl и TWinControl?

27 Известно, что некоторый класс T является потомком класса TWinControl. Что можно сказать о свойствах и “поведении” объектов класса T?

28 Что означает директива DYNAMIC?

29 Что означает директива __fastcall?

30 Как следует перекрывать метод SetFocus в потомках класса TWinControl?

31 Что общего и каковы отличия методов KeyDown и KeyPress класса TWinControl?

Лабораторное задание

Построить класс – потомок одного из классов стандартных визуальных компонентов. Название компонента - родителя и отличительное свойство класса-потомка указаны в табл. Л.5.1.

Таблица Л.5.1 − Варианты индивидуальных заданий

№ вар	Компонент - родитель	Отличительное свойство компонента-потомка
1	TLabel	Объекты компонента-потомка при щелчке правой кнопкой мыши увеличивают размер букв своего Caption вдвое, а при повторном щелчке правой – восстанавливают предыдущие размеры букв
2	TStringGrid	Объект компонента-потомка при нажатии клавиши F11 выводит в отдельном окне сообщение о своих параметрах: Name, координаты размещения, количество строк и столбцов
3	TMemo	Объекты компонента-потомка при нажатии клавиши F11 увеличивают свой размер на 50 пикселей в высоту и на 75 – в ширину, при нажатии F12 – восстанавливают предыдущие размеры
4	TLabel	Объекты компонента-потомка при щелчке правой кнопкой мыши выполняют “обмен местами” пары цветов [цвет фона – цвет букв Caption], а при повторном щелчке правой кнопкой – восстанавливают предыдущие цвета
5	TEdit	Объекты компонента-потомка при нажатии клавиши ENTER перемещается в верхний левый угол формы, а при повторном нажатии – возвращается на старое место
6	TButton	Объект компонента-потомка при нажатии (щелчке левой кнопкой мыши) выдает столько звуковых сигналов, сколько всего было нажатий данного объекта
7	TMemo	Объект компонента-потомка при нажатии клавиши F11 выводит сообщение о том, вводился ли когда-нибудь в его окно какой-либо текст, который содержал слово ОБЪЕКТ
8	TStringGrid	Объект компонента-потомка при нажатии клавиши F11 выдает столько звуковых сигналов, сколько есть заполненных ячеек (Cells) в окне данного объекта
9	TEdit	Объект компонента-потомка при нажатии клавиши ENTER выводит сообщение о длине максимально длинного текста, который когда-либо находился в его окне

Продолжение таблицы Л.5.1

№ вар	Компонент - родитель	Отличительное свойство компонента-потомка
10	TMemo	У объектов компонента-потомка при нажатии клавиши F11 появляются “бегунки” (ScrollBars), а при повторном нажатии F11 “бегунки” исчезают
11	TButton	При нажатии (щелчке левой кнопкой мыши) увеличивает свой размер на 15 пикселей в высоту и на 20 – в ширину; а при повторном нажатии – восстанавливает предыдущие размеры
12	TLabel	Объекты компонента-потомка при щелчке правой кнопкой мыши исчезают на 1,5 секунды, а потом снова появляются
13	TLabel	Объект компонента-потомка при щелчке правой кнопкой мыши выводит в отдельном окне сообщение о своих параметрах: Name, Caption, координаты размещения и количество символов в Caption
14	TEdit	Объект компонента-потомка при нажатии клавиши ENTER изменяет все буквы своего Text на большие, а при повторном нажатии – восстанавливает предыдущий размер букв
15	TStringGrid	У объектов компонента-потомка при нажатии клавиши F11 исчезают “бегунки” (ScrollBars), а при повторном нажатии F11 “бегунки” появляются
16	TMemo	Объект компонента-потомка при нажатии клавиши F11 выдает столько звуковых сигналов, сколько всего имеется строк в окне данного объекта
17	TStringGrid	У объектов компонента-потомка при нажатии клавиши F11 исчезают фиксированные строка и столбец (FixedRow и FixedCol), при повторном нажатии F11 они снова появляются
18	TButton	Объект компонента-потомка при нажатии (щелчке левой кнопкой мыши) выводит в отдельном окне сообщение о своих параметрах: Name, Caption, координаты размещения и число нажатий
19	TEdit	Объекты компонента-потомка при нажатии клавиши ENTER выполняют “обмен местами” пары цветов [цвет фона – цвет букв], а при повторном нажатии восстанавливают предыдущие цвета
20	TMemo	Объект компонента-потомка при нажатии клавиши F11 выводит сообщение о длине максимально длинного текста, который когда-либо находился в его окне
21	TStringGrid	У объектов компонента-потомка при нажатии клавиши F11 все пустые ячейки Cells заполняются строкой из трех символов ###, а при повторном нажатии F11 эти ячейки снова очищаются

Продолжение таблицы Л.5.1

№ вар	Компонент - родитель	Отличительное свойство компонента-потомка
22	TLabel	Объект компонента-потомка при щелчке правой кнопкой мыши изменяет все буквы своего Caption на заглавные, а при повторном щелчке правой кнопкой – восстанавливает предыдущий размер букв
23	TEdit	Объект компонента-потомка при нажатии клавиши ENTER выводит сообщение о том, вводилась ли когда-либо какая-нибудь текстовая информация в его окно
24	TStringGrid	У объектов компонента-потомка при нажатии клавиши ESC все ячейки Cells очищаются
25	TStringGrid	Объекты компонента-потомка при нажатии клавиши F11 транспонируют матрицу в своем окне (меняют местами строки с колонками в Cells)
26	TMemo	У объектов компонента-потомка, кроме стандартного окна для введения текста, присутствует еще и окно (или надпись), в котором постоянно отображается количество символов текста
27	TButton	Объекты компонента-потомка при нажатии (щелчке левой кнопкой мыши) исчезают на одну секунду, а потом снова появляются

5 Виртуальные и динамические методы. Полиморфизм

5.1 Построение класса “1-го уровня”

Построим класс, каждый объект которого содержит информацию о доходах некоторого гражданина, а методы предназначены для вычисления налогов этого гражданина. Мы предположим, что класс включает следующие поля:

fPerson – фамилия гражданина (до 20 символов);

fBirth – год рождения гражданина (положительное целое число длиной 2 байта);

fAddress – адрес проживания гражданина (до 150 символов);

fCurrYear – год, для которого вычисляются налоги (положительное целое длиной 2 байта);

fIncomes – доходы гражданина в году fCurrYear (величина типа float).

Необходимо определить конструктор класса для обеспечения возможности задавать конкретные значения полей. Кроме этого, объявим такие методы класса:

CalcTax – функция, которая вычисляет размер налога;

GetInfo – функция, которая возвращает текстовую строку с информацией о гражданине, с суммой налога включительно.

Будем считать, что действуют такие правила начисления налога:

1 с лиц моложе 17-ти лет налог не взимается;

2 если величина прибыли fIncomes меньше 100, налог не взимается;

3 если прибыль fIncomes находится в промежутке 100 ... 10 000, налог составляет 20 % от величины (fIncomes – 100);

4 если Incomes больше 10 000, то для его части, равной 10 000, налог исчисляется по правилу 3, а из остатка (fIncomes-10 000) взимается 60 % .

Этих сведений достаточно для построения нашего класса. Назовем этот класс **TPerson**. Далее приводится текст соответствующей библиотеки.

Файл MyClass1.h :

```
#ifndef MyClass1H
#define MyClass1H
//---------------------------------------------
const __int8 Child = 17;
const  float Coef1 = 0.2,
             Coef2 = 0.6;
typedef char s20[21];
typedef char s150[151];
typedef unsigned __int16 YearNumb;
class TPerson{
private:
        s20     fPerson;
        YearNumb   fBirth;
        s150   fAddress;
```

```
        YearNumb   fCurrYear;
        Float  fIncomes;

protected:
        float CalcTax(void);
public:
        TPerson( s20  iPerson,          // constructor
                 YearNumb  iBirth,
                  s150   iAddr,
                  YearNumb   iCurr,
                   float   iIncomes);
        ~TPerson(void);                 // destructor
        AnsiString GetInfo(void);
};
#endif
```

В начале файла заголовков объявлены константы Child – возраст, начиная с которого взимается налог, а также Coef1 и Coef2 – коэффициенты, употребляемые при вычислении налога. После этого приведены объявления типов для имени гражданина, его адреса, а также тип для значений года его рождения и текущего года.

Созданный нами класс не имеет родительского класса, поэтому непосредственно за именем TPerson приведен перечень его полей и методов. Поскольку описание полей объектов класса TPerson находится в секции private, у пользователя отсутствует прямой доступ к их значениям. Для изменения любого из полей (fPerson, fBirth и др.) необходимо воспользоваться открытым (public) конструктором TPerson, задав соответствующие значения его аргументов. Обратите внимание: тип функции-конструктора перед именем TPerson задавать не следует (в том числе и тип void не является здесь допустимым).

Открытыми методами являются также:

функция-деструктор ~TPerson экземпляров класса TPerson, и

функция GetInfo, возвращающая строку с информацией о данном объекте т.е. о данном гражданине).

Секция protected нашего класса содержит объявление метода CalcTax. Это означает, что CalcTax является недоступным для вызова из программ пользователя; кроме одного исключения: если пользователь перекроет какой-либо из методов класса TPerson, то в тексте этого перекрытия он имеет возможность вызвать CalcTax (немного позже мы так и сделаем).

Объявление класса заканчивается символом « **;** ».

Теперь рассмотрим **файл MyClass1.cpp:**

```
#include <vcl.h>
#pragma hdrstop
#include "MyClass1.h"
#pragma package(smart_init)
TPerson::TPerson( s20  iPerson,          //реализация конструктора
```

```
                        YearNumb  iBirth,
                        s150  iAddr,
                        YearNumb  iCurr,
                        float  iIncomes)
{ strcpy(fPerson, iPerson);
  fBirth=iBirth;
  strcpy(fAddress,iAddr);
  fCurrYear=iCurr;
  fIncomes=iIncomes;
}
TPerson::~TPerson(void){ Beep(300,1000); }
float TPerson::CalcTax(void)
{
if( (fCurrYear-fBirth<=Child) || (fIncomes<100) ) return 0;
else
if( fIncomes<10000 ) return Coef1*(fIncomes-100);
else
   return Coef1*9900+ Coef2*(fIncomes-10000);
}
AnsiString TPerson::GetInfo(void)
{
float tax;
AnsiString s;
tax=CalcTax( );
s=AnsiString(fPerson)+"  "
  +IntToStr(fBirth)+"  "
  +fAddress+"  "
  +IntToStr(fCurrYear)+"  "
  +FloatToStr(fIncomes, ffFixed, 7,2)+"  "
  +FloatToStr(tax, ffFixed, 7,2);
return s;
}
```

В теле конструктора значения аргументов, указанные в его заголовке, просто переписываются в поля объекта-экземпляра класса. Можно добавить к методам класса TPerson еще один конструктор, который будет строить объекты-экземпляры по другому, например, на основании информации из файла:

```
TPerson::TPerson( AnsiString FileName, int FilePos ){ ... }
```

где параметрами будут имя файла и номер записи в файле, в котором содержатся сведения о данном гражданине.

Далее, как видите, функция-деструктор ~TPerson в нашем примере не выполняет ничего, кроме выдачи звукового сигнала.

Приведем пример программы, в которой используется построенный класс TPerson. В ней объявлен экземпляр P_1 этого класса с инициализацией его с помощью конструктора. Далее вызвана функция P_1.GetInfo, которая формирует string-строку для вывода в окно Memo1.

```
#include "MyClass1.h"
void __fastcall TForm1::Button1Click(TObject *Sender)
{
TPerson  P_1("Япончик", 1891, "Одесса", 2007, 1000000);
AnsiString s;
s=P_1.GetInfo( );
Memo1->Lines->Add(s);
}
```

Обратите внимание на то, что объявление объекта нашего класса обязательно должно сопровождаться списком инициализации его параметров в круглых скобках. Это необходимо потому, что мы не предусмотрели никакого конструктора, который разрешал бы инициализировать поля создаваемых объектов “по умолчанию”.

Обратите внимание также на то, что, поскольку после завершения работы функции TForm1::Button1Click память для всех ее локальных имен освобождается, будет “уничтожен” и объект P_1. Поэтому будет вызван деструктор этого объекта, хотя явного его вызова в тексте программы нет. Вы должны услышать звуковой сигнал, предусмотренный текстом деструктора нашего класса (именно для этого деструктор “по умолчанию” и был нами перекрыт).

5.2 Назначение секции protected

Объявив класс TPerson в модуле MyClass1, мы можем строить потомки этого класса в любом модуле, который ссылается на MyClass1. При этом C++-текст
в файле MyClass1.cpp может быть для нас недоступен.

Построим новый модуль MyClass2 и поместим в него потомок TPerson2 класса TPerson, объекты которого будут иметь дополнительное поле fPriviledge типа bool. Это поле определяет “льготы” гражданина. Если fPriviledge=false, начисление налога для объекта (гражданина) класса-потомка выполняется так же, как в классе MyClass1. Однако, если fPriviledge=true, налог такого гражданина следует уменьшить вдвое.

Текст **файла MyClass2.h :**

```
#ifndef MyClass2H
#define MyClass2H
#include "MyClass1.h"
class TPerson2 : public TPerson {
private:
        bool  fPriviledge;
protected:
        float  CalcTax(void);
```

```
public:
        TPerson2( s20  iPerson,          // constructor
                  YearNumb  iBirth,
                   s150  iAddr,
                  YearNumb  iCurr,
                   float  iIncomes,
                   bool  iPriv);
        ~TPerson2(void);                 // destructor
};
//---------------------------------------------
#endif
```

Новое поле помещено в секцию private, и оно будет “прямо” недоступным вне MyClass2 (так же, как поля fPerson, fBirth и др. недоступны вне MyClass1). Однако эти поля наследуются и их инициализацию можно выполнить с помощью конструктора TPerson2.

Функция-метод CalcTax переобъявлена в секции protected, поскольку она будет перекрываться в классе TPerson2 (всё, что перекрывается, следует переобъявлять).

Как видите, мы объявили также деструктор нового класса.

Текст **файла MyClass2.cpp**:

```
#include <vcl.h>
#pragma hdrstop
#include "MyClass2.h"
#pragma package(smart_init)
TPerson2::TPerson2( s20  iPerson,            // constructor
                    YearNumb iBirth,
                     s150 iAddr,
                     YearNumb iCurr,
                     float iIncomes,
                     bool iPriv)
          : TPerson( iPerson, iBirth, iAddr, iCurr, iIncomes)
{
fPriviledge = iPriv;
}
TPerson2::~TPerson2(void){ Beep(1000,500); }
float TPerson2::CalcTax(void)
{
float tax=TPerson::CalcTax( );
if(fPriviledge)tax=0.5*tax;
return tax;
}
```

Обратите внимание, в первую очередь, на вызов родительского конструктора, который стоит за двоеточием после заголовка конструктора TPerson2. По-

мните: родительский конструктор не наследуется и его нужно в необходимых случаях вызывать явным образом.

В данном случае инициализация “родительских” полей осуществляется через вызов родительского метода TPerson. А в теле нового конструктора TPerson2 происходит инициализация нового поля.

В отличие от конструктора, родительские деструкторы всех “предков” наследуются всеми их классами-потомками. Во время уничтожения объекта-потомка класса сначала вызывается его собственный деструктор, а потом – деструктор родительского класса. Поэтому наш новый деструктор ~TPerson2, фактически, просто добавляет новые звуки к звучанию ~TPerson (это даст возможность различать по звуку, объект какого именно класса уничтожается).

Метод CalcTax, который перекрывается в потомке, сначала вызывает свой унаследованный метод (для вычисления “старого” налога), а потом добавляет необходимые новому объекту операции.

Как видим, унаследованный protected-метод CalcTax без помех вызывается в тексте перекрытого метода CalcTax. Так же его можно было бы вызвать в тексте TPerson2 или ~TPerson2. Тем не менее, если вы попробуете вызвать CalcTax не в методе класса-потомка, возникнет ошибка: Undeclared identifier: 'CalcTax'. Именно в этом и заключается “защита” элементов класса, объявленных в protected-секции.

Что касается метода GetInfo, то он наследуется классом TPerson2 от TPerson и, поскольку кажется нецелесообразным что-то в нем изменять, не станем перекрывать его в TPerson2.

Приведем пример программы, которая работает одновременно с объектом класса TPerson и с объектом класса-потомка TPerson2.

```
#include "MyClass1.h"
#include "MyClass2.h"
void __fastcall TForm1::Button1Click(TObject *Sender)
{
TPerson  P_1("Япончик", 1891, "Одесса", 2007, 1000000);
TPerson2 P_2("Сонька-Золотая ручка", 1886, "Одесса", 2007, 1000000,
true);
AnsiString s;
s=P_1.GetInfo( );
Memo1->Lines->Add(s);
s=P_2.GetInfo( );
Memo1->Lines->Add(s);
}
```

5.3 Полиморфизм

К сожалению, выполнив предыдущую программу, мы увидим, что выведенные в Memo1 значения налога для Япончика и для Соньки будут тождественными. Как же так? Ведь Япончик является объектом типа TPerson, который не имеет поля Priviledge, в то время как Сонька имеет тип TPerson2 и true в

поле Priviledge. Итак, поскольку доходы Япончика и Соньки одинаковы, налог с последней должен быть вдвое меньшим!

Загвоздка – в методе GetInfo. Напомним, что в тексте этого метода класса TPerson вызывается функция CalcTax:

```
AnsiString TPerson::GetInfo(void)
{
float tax;
...
tax=CalcTax( );   //вызывается метод класса TPerson !
...
}
```

Функция CalcTax – это метод класса TPerson. Хотя мы и перекрыли CalcTax в классе-потомке TPerson2, унаследованная от “старого” класса функция GetInfo вызывает метод “своего” класса, а не класса-потомка. А в старом методе, естественно, никак не учтено Priviledge.

Чтобы исправить положение, можно перекрыть метод GetInfo в классе TPerson2. При этом нет необходимости изменять хотя бы одну букву в тексте GetInfo: достаточно дописать заголовок GetInfo в объявлении класса (файл MyClass2.h) и повторить его текст в файле MyClass2.cpp. Во время трансляции этого текста в вызове tax=CalcTax( ) будет использован уже адрес нового, перекрытого метода CalcTax.

Этот способ неудовлетворителен. Не очень приятно переписывать тождественные тексты. Но главное, – чтобы переписать, нужно знать этот текст, то есть файл MyClass1.cpp следует сделать открытым для пользователя. А ведь этот файл введен, в частности, и для того, чтобы тексты программ – предмет know how – были недоступными.

С другой стороны, во время выполнения нашей программы Button1Click известно все, что необходимо для правильных обращений к CalcTax в различных вызовах GetInfo. Действительно, если GetInfo вызывается объектом P_1, а этот объект принадлежит к типу TPerson, то в тексте GetInfo следует обратиться к CalcTax, которая “присуща” классу TPerson. Если же вызов выполняет объект P_2 типа TPerson2, процедура GetInfo должна обратиться к CalcTax из класса TPerson2. Это означает, что адрес процедуры, которую транслятор подставляет вместо слова CalcTax в операторе tax=CalcTax, должен изменяться во время выполнения программы в зависимости от того, какой именно объект вызывает GetInfo.

Как правило, во время трансляции C++-модулей вместо вызовов подпрограмм транслятор подставляет адреса-константы. Чтобы принудить его оставить данный адрес нереализованным, и вычислять его в момент обращения к данной подпрограмме во время выполнения модуля, соответствующую подпрограмму следует объявить динамической, или же виртуальной. Для этого надо дописать директиву DYNAMIC или директиву virtual к заголовку метода в объявлении класса, как в рассмотренных ранее примерах методов:

```
DYNAMIC  void __fastcall Click(void);
virtual  void __fastcall SetFocus(void);
```

Итак, нам нужно немного изменить прототип метода CalcTax в объявлении класса TPerson (файл MyClass1.h):

```
class TPerson{
...
protected:
        virtual  float  CalcTax(void);
...
}
```

При этом дописывать оператор virtual в заголовок метода CalcTax в файле MyClass1.cpp не следует.

Что касается модуля MyClass2, то его текст можно вовсе не менять. Дело в том, что если метод родительского класса объявлен виртуальным (или динамическим), он останется таким во всех потомках этого класса.

Поскольку при уничтожении объекта необходимо, чтобы вызывался деструктор класса именно этого объекта, а не только его родительского класса, деструкторы, как правило, объявляют виртуальными. Это предотвращает ошибочное освобождение памяти, захваченной для полей-указателей объекта. Хотя у нас таких полей нет, целесообразно соблюдать правила, объявив (в файле MyClass1.h):

```
virtual  ~TPerson(void);
```

Если внести эти два изменения в текст объявления класса TPerson, то, выполнив программу Button1Click, можно убедиться, что налог Соньки теперь стал вдвое меньше, чем налог Япончика.

Обратите внимание на отсутствие потребности объявления виртуальной процедуры GetInfo. В тексте этой процедуры есть вызов виртуального метода (CalcTax), и адрес в этом вызове вычисляется во время выполнения программы. Тем не менее, адрес самого метода GetInfo не изменяется: все вызовы GetInfo указывают на один и тот же код.

Для методов, которые не являются динамическими или виртуальными, употребляют термин невиртуальные. GetInfo является примером невиртуального метода.

В отличие от невиртуального, параметры виртуального или динамического метода при перекрытии должны быть целиком тождественны родительским.

Полиморфизмом называется способность программного кода обрабатывать структуры, точное описание которых неизвестно в момент трансляции этого кода.

Рассмотренная выше программа GetInfo является примером полиморфного кода. Ведь в ней имеется вызов подпрограммы CalcTax, точное описание которой неизвестно в момент трансляции GetInfo. CalcTax вычисляет некоторую float-величину, но какую именно (или: как именно)? Возможность перекрытия CalcTax влечет отсутствие ответа на этот вопрос.

Если метод M объявлен виртуальным и он вызывается в некотором другом методе F того же самого класса K, то, перекрыв метод M в потомке K, мы в определенном смысле “вставляем свой собственный C++-код” (код метода M) в

текст метода F. При этом нам не нужно знать текст F: чтобы работать с полиморфными программами, необязательно углубляться в детали их внутреннего строения.

Полиморфизм дает возможность дописывать и переписывать разработанные прежде методы, сохраняя при этом прежние связи с другими методами (а именно: “новый” метод CalcTax “по-старому” взаимодействует со “старым” методом GetInfo). Поэтому расширять возможности многих программных систем, созданных для работы под управлением WINDOWS, можно, не изучая подробно безграничного объема сведений, касающихся деталей WINDOWS.

Лабораторная работа № 6

Виртуальные и динамические методы. Полиморфизм

Цель работы: Научиться строить классы, которые не являются потомками визуальных компонентов C++-Builder. Ознакомиться с понятием полиморфизма.

Контрольные вопросы

1 Каковы функции метода-конструктора?

2 Что такое деструктор?

3 Почему деструктор в составе класса TObject библиотеки VCL объявлен как virtual?

4 Почему функция-метод CalcTax класса TPerson (в библиотеке MyClass1) не имеет формальных параметров? И почему их имеет конструктор того же класса?

5 Почему все поля класса TPerson (модуль MyClass1) помещены в секцию private?

6 Пусть мы пользуемся модулем MyClass1, объявили и произвели инициализацию объекта P_1 типа TPerson, а немного позже нам потребовалось изменить значение поля Incomes этого объекта. Как это сделать?

7 Постройте конструктор для класса TPerson, который выполнял бы инициализацию экземпляров этого класса, считывая значения их полей из файла.

8 Объясните, почему для класса TPerson “не очень нужно” перекрытие метода-деструктора.

9 Метод M класса K не перекрывается в потомке K1 класса K. Какие ссылки на M возможны в модуле, где объявлен потомок K1?

10 Метод M класса K перекрывается в потомке K1 класса K. Какие ссылки на M должны появиться в модуле, где объявлен потомок K1?

11 Один и тот же метод M класса K перекрывается в разных классах-потомках K1 и K2. При этом в классе K1 в теле перекрытого метода M стоит вызов

```
K::M( );
```

а в классе K2 в теле перекрытого M стоит

```
K::M(x,y);
```

Объясните возможную причину разной формы приведенных обращений к унаследованному M.

12 Метод M объявлен в секции private класса K. Объект Z является экземпляром класса K. Допустим ли вызов Z.M ?

13 Метод M объявлен в секции protected класса K. Объект Z является экземпляром класса K. Допустим ли вызов Z.M ?

14 Метод M объявлен в секции protected класса K. Метод P – это второй метод класса K. Допустим ли вызов Z в теле P?

15 Метод M объявлен в секции protected класса K. Метод P объявлен в классе K1, который является потомком класса K. Допустим ли вызов Z в теле P?

16 Объясните, почему в “первом варианте” объявления класса-потомка TPerson2 значения налога для объектов “Япончик” и “Сонька” (отображенные в

окне Memo1) оказались тождественными. Можно ли утверждать, что метод CalcTax объекта “Сонька” ошибочно вычисляет налог?

17 В чём заключается основной недостаток идеи, согласно которой в случае перекрытия некоторого метода M следует перекрыть также все методы, которые содержат ссылку на M? И каковы “второстепенные” недостатки этой идеи?

18 Что следует знать, чтобы вычислить правильный адрес метода CalcTax в его вызове из процедуры GetInfo?

19 Объясните, что именно происходит в случае, когда к заголовку метода дописывается директива virtual?

20 Чем отличаются директивы DYNAMIC и virtual?

21 Верно ли, что, если в объявлении класса заменить всюду директиву DYNAMIC директивой virtual, видимых изменений в поведении объектов класса мы не увидим? А если увидим, то какие именно?

22 Можно ли вызвать унаследованный метод в перекрытом невиртуальном методе?

23 Можно ли вызвать унаследованный метод в перекрытом динамическом методе?

24 Можно ли перекрыть невиртуальный метод, объявив результат этого перекрытия виртуальным?

25 Можно ли перекрыть виртуальный метод, объявив результат этого перекрытия невиртуальным?

26 Метод M никогда не будет перекрываться. Нужно ли объявить его в таком случае невиртуальным? Объясните почему.

27 Метод M и его перекрытие в потомках могут вызываться объектами, тем не менее не вызываются никакими *методами* объектов. Нужно ли объявить M виртуальным? Объясните почему.

28 Что такое полиморфизм? Приведите примеры.

29 В каком смысле вы “вставляете” свой код “в середину” полиморфного объекта?

30 Каковы основные преимущества объектного программирования? Верно ли,
что любую задачу можно запрограммировать, не обращаясь к объектам?

Лабораторное задание

Все представленные в табл. Л.6.1 задачи предусматривают построение класса 1-го уровня, одним из элементов которого будет функция, определяющая “качество” элемента (качество оценивается действительным числом Q). Кроме того, следует построить класс-потомок первого класса, который содержит дополнительное поле P и перекрывает функцию качества, учитывая значение этого поля. В отдельной программе следует объявить объекты как первого, так и второго класса, инициализировать их и вывести информацию о них в окно (окна) формы.

В табл. Л.6.1 для класса 1-го уровня указаны название, поля и правило вычисления качества Q объектов класса; для класса-потомка указаны дополнительное поле P и модифицированное правило качества Qp.

Таблица Л.6.1 − Варианты индивидуальных заданий

№	Класс 1-го уровня	Класс-потомок
1	Компьютер: тактовая частота процессора (МГц) объем оперативной памяти (Мб) Q = (0,1*частота) + память	P: объем винчестера (Гб) Qp=Q+0,5*P
2	Армия: численность (тыс. чел.) вооруженность (в баллах − float-число) Q = 0,3*численность+0,7*вооруженность	P: опыт (число месяцев, на протяжении которых армия вела боевые действия) Qp = Q * (P+1)
3	Автомобиль: мощность двигателя (кВт) количество мест Q = (0,1*мощность)+количество мест	P: год изготовления Qp=Q-1,5*(T-P), где T − текущий год
4	Человек Толстого: самооценка (в баллах - целое число) оценка другими людьми (в баллах) Q = (оценка другими)/самооценка	P: оценка потомками (в баллах) Qp = 0,3*Q + 0,7*P
5	Телевизор: диагональ экрана (см) звуковая мощность (дБ) Q = диагональ+(0,05*мощность)	P: страна-производитель Qp: Если страна − Япония, то Qp=2*Q; а если Сингапур или Корея, то Qp=1,5*Q
6	Мушкетер: количество дуэлей количество любовниц Q = 2*дуэли + любовницы	P: количество бутылок, которые может выпить за один раз Qp = Q + 0,5*P
7	Экзамен: количество студентов на экзамене продолжительность экзамена (ч.) Q = количество студентов /продолжительность	P: процент двоек (1, 2, ..., 100) Qp = Q*(100-P)/100
8	Солдат: рост (м) вес (кг) Q = рост*вес	P: образование (начальное, среднее, высшее) Qp: если образование высшее, то Qp=2*Q; а если начальное, то Qp=0,5*Q
9	Компьютерная сеть: количество рабочих станций среднее расстояние между станциями (м) Q = (количество станций)*(среднее расстояние)	P: средняя скорость передачи данных в сети (Мб/с) Qp = Q*P

Продолжение таблицы Л.6.1

№	Класс 1-го уровня	Класс-потомок
10	Полководец: количество битв количество побед Q = (количество побед)^2/количество битв	P: количество побед с меньшими, чем у противника, силами Qp = P^2/количество битв +Q
11	Дом: количество комнат год сооружения Q=(количество комн.)-2*(T-год сооруж.) где T − текущий год	P: район (центр города, окраина, и т.п.) Qp: если район центральный, то Qp=2*Q; а если окраина, то Qp=0,5*Q
12	Королева красоты: оценка красоты в баллах (float-число) рост (см) Q = красота+0,5*рост	P: коэффициент интеллекта Qp = Q + 0,01*P
13	Коньяк: крепость (градусы) выдержка (звездочки) Q = количество звездочек + крепость/100	P: страна-производитель Qp: если страна − Франция, то Qp=Q+1, а если Грузия, Армения или Молдова, то Qp=Q+0,5
14	Программист: количество написанных программ количество языков программирования, кото-рыми владеет Q = (количество программ)*(количество языков)	P: количество программ, которые работают без ошибок Qp = Q * P / (количество всех программ)
15	Партия: численность (тыс. членов) процент голосов на последних выборах Q = 0,3*численность+0,7*процент	P: увеличилась или сократилась за последний год Qp: если увеличилась, то Qp=1,2*Q; а если сократилась, то Qp=0,8*Q
16	Броненосец (линейный корабль): главный калибр (дюймы) количество пушек главного калибра Q = (количество пушек)*калибр	P: крейсерская скорость (в морских узлах) Qp = 0,25*Q + P
17	Спортсмен: количество соревнований сумма мест, занятых в соревнованиях Q = (количество соревнований) / (сумма мест)	P: занимал ли хотя бы один раз первое место Qp: Если P, то Qp = 1,5*Q

Продолжение таблицы Л.6.1

№	Класс 1-го уровня	Класс-потомок
18	Высшее учебное заведение: прием на 1-й курс (чел.) выпуск с последнего курса (чел.) Q = выпуск / прием	P: процент выпускников, которые работают по специальности Qp = P*Q
19	Жених: возраст (годы) красота (в баллах − float-число) Q = красота – (возраст–20)^2	P: благосостояние (тыс. $) Qp = P^3 + Q
20	Алмаз: вес (в каратах) качество огранки в баллах (float-число) Q = 0,4*вес + 0,6*огранка	P: цвет (белый, голубой, желтый, и т.п.) Qp: если цвет голубой, то Qp=Q+1; а если желтый, то Qp=Q–0,5
21	Коробок спичек: количество спичек в коробке время горения одной спички (с) Q = (количество спичек)*время	P: средний процент бракованных спичек в коробке Qp = (100-P) * Q / 100
22	Водка: крепость (градусы) качество сырья (действительное число) Q = качество сырья + крепость/100	P: на чем настоена (поле может быть пустым) Qp: если настоена на калгане, то Qp=Q+0,5, если на чём-то другом, то Qp=Q+0,25
23	Митинг: n1 = количество ораторов n2 = количество участников Q = (n2-n1)*n2	P: количество групп ораторов, которые высказывали одинаковые мысли Qp = Q * P
24	Попугай: возраст (годы) размеры (в баллах - float-число) Q = возраст * размеры	P: умение говорить Qp: если P имеет место, то Qp = 10*Q; иначе Qp = Q
25	Спектакль: n1 = количество зрителей в начале n2 = количество зрителей в конце Q = n2 / n1	P: год написания пьесы Qp = Q * (T-P+1) где T − текущий год
26	Врач: суммарное количество пациентов суммарное количество правильных диагнозов Q=количество правильных диагнозов / количество пациентов	P: количество лиц, которые после выздоровления и нового заболевания снова обратились к данному врачу Qp = Q * P
27	Демонстрация: количество участников количество милиционеров Q = участники*милиционеры	P: разрешенная или нет Qp: если разрешенная, то Qp = Q; иначе Qp = 10*Q

Литература

1 **Леонов Ю.Г., Силкина Н.В., Шпинова О.Д.** Программирование на алгоритмическом языке С++. Учеб. пособие с лабораторным практикумом.
– Одесса: ОНАС, 2002. – 68 с.

2 **Леонов Ю.Г., Угрік Л.М., Швайко І.Г.** Збірник задач з програмування. – Одеса: УДАЗ, 1997. – 80 с.

3 **Архангельский А.Я., Тагин М.А.** Программирование в С++ Builder 6 и 2006. – М.: Бином-Пресс, 2007. – 1184 с.

4 **Архангельский А.Я.** Программирование в С++ Builder 6. – М.: Бином-Пресс, 2002. – 527 с.

5 **Березин Б.Н., Березин С.Б.** Начальный курс С и С++. – М.: Диалог-МИФИ, 2000. – 288 с.

6 **Страуструп Б.** Язык программирования С++. – СПб.-М.: Бином, 1999. – 991 с.

7 **Калверт Чарльз, Рейдорф Кент.** Borland C++ Builder 5. Энциклопедия программиста. К.: Издательство «Диасофт», 2001, - 944 с.

8 **Бобровский С.** Самоучитель программирования на языке С++ в системе Borland C++ Builder 5.0. М: - Десс Ком, 2001.

9 **Культин Н.Б.** С++ Builder в задачах и примерах. СП.: БХВ – Петербург, 2005. – 336 с.

Редактор И. В. Ращупкина
Компьютерное макетирования Ж.А. Гардыман