Министерство транспорта и связи Украины
ОДЕССКАЯ НАЦИОНАЛЬНАЯ АКАДЕМИЯ СВЯЗИ им. А.С.ПОПОВА

Кафедра информационных технологий

Буката Л.Н., Кузнецов В.Д.

Информатика
Модуль 1
Основные сведения о персональном компьютере. Организация вычислительных процессов с линейной и разветвленными структурами

Часть 1

Учебное пособие
для студентов всех специальностей академии

Одесса 2007

УДК 004.43 План УМИ 2007 г.

Учебное пособие разработали:
Л. Н. Буката
В. Д. Кузнецов

Учебное пособие рассмотрено и одобрено к изданию на заседании кафедры ИТ.
Протокол № 4 от 12 декабря 2007 г.

Зав. кафедрой Леонов Ю.Г.

Учебное пособие рассмотрено и одобрено методическим советом факультета ИС.
Протокол № 15 от 5 июля 2007г.

Декан Факультета ИС И.В. Стрелковская

СОДЕРЖАНИЕ

Введение

Учебная дисциплина “Информатика”

Дисциплина “Информатика”**,** изучаемая в I-II семестрах первого курса на всех факультетах академии, предназначена для обучения студентов работе на персональном компьютере с целью его использования в своей настоящей и будущей профессиональной деятельности.

Цель курса – формирование у студентов знаний и навыков в таких областях, как:

архитектура компьютера;

работа в среде операционной системы Windows;

алгоритмизация вычислительных процессов;

составление программ на алгоритмическом языке C++;

знакомство с объектно-ориентированным программированием на примере решения самых простых задач в среде программирования C++ Builder.

Программа курса состоит из четырех модулей:

модуль 1 – основные сведения о персональном компьютере и организация вычислительных процессов с линейной и разветвленными структурами;

модуль 2 – программирование на алгоритмическом языке C++ задач с циклами и массивами;

модуль 3 – программирование задач со структурированными типами данных;

модуль 4 – изучение основ объектно-ориентированного программирования.

Структура модуля 1

Место в структурно-логической схеме дисциплины **“**Информатика”**:** Исходное: она предшествует изучению всех профессионально ориентированных дисциплин.

Базируется на школьном курсе информатики (студенты должны владеть знаниями в объеме школьного курса информатики согласно программе Министерства образования) и опирается на школьный курс математики и дисциплину «Высшая математика» (функции, формулы преобразования, ряды). Кроме лекций и упражнений, Программа представленного в пособии модуля 1 предусматривает лабораторный практикум и выполнение комплексного задания «Программирование линейных и разветвленных процессов».

Перечень лабораторных работ и практических занятий

Приблизительный перечень лабораторных работ

Лабораторная работа № 1. Демонстрационная работа. Создание папки. Создание файла документа. Работа со "стандартными" программами Windows. Элементы работы в операционной системе Windows: работа с ярлыками, папками, меню. Работа с файлами.

Лабораторная работа № 2. Создание и форматирование текста в редакторе Word.

Лабораторная работа № 3. Создание и обработка таблиц в Word. Графические изображения и редактор формул в Word. Выполнение схем алгоритмов и описание задания в Word.

Лабораторная работа № 4. Создание простейших проектов в C++ Builder (калькулятор, изменение цвета формы и др.).

Лабораторная работа № 5. Создание проекта программы линейной структуры и выполнение программы на ЭВМ.

Лабораторная работа № 6. Линейные структуры, организация ввода данных и вывода результатов в компоненты Edit.

Лабораторная работа № 7. Выполнение проекта программы с разветвлением. Вывод данных в компоненты Мемо, ListBox.

Лабораторная работа № 8. Выполнение проекта программы с оператором вариантов switch и выводом информационных сообщений.

Перечень практических занятий

Практическое занятие № 1. Двоичная, восьмеричная и шестнадцатиричная системы счисления. Операционная система Windows. Основные принципы работы в системе. Доступ к дискам, папкам и файлам. Создание, поиск, перемещение и удаление папок. Выполнение программ. Основные сведения о текстовом редакторе Word. Создание текста.

Практическое занятие № 2. Сведения о текстовом редакторе Word. Главное меню. Окно редактора. Панель инструментов. Элементы форматирования документа. Выделение фрагмента текста и работа с ним, открытие и сохранение документа. Создание и обработка таблиц в Word. Сортировка данных. Построение графиков. Назначение и построение экрана редактора формул Word. Ввод формул и текста (комментарий). Форматирование и редактирование формул. Рисование в Word.

Практическое занятие № 3. C++ Builder – среда визуального программирования: главное меню, локальное меню, панель быстрого доступа, палитра компонентов, форма и редактор кода, инспектор объектов. Компоненты Button, Panel, Label. Последовательность создания проекта в C++ Builder. Примеры создания самых простых демонстрационных проектов в C++ Builder .

Практическое занятие № 4. Организация линейных процессов. Запись арифметических выражений. Оператор присваивания. Ввод и вывод данных в компонент Edit. Примеры составления программ с линейной структурой. Выдача комплексного задания по теме: "Составление алгоритмов и программ с линейной и разветвленной структурами".

Практическое занятие № 5. Пример создания проекта с реализацией программы линейной структуры. Контрольная работа № 1 по теме "Линейные процессы".

Практическое занятие № 6. Создание проекта программы с условными операторами. Компонент Мемо. Информационные сообщения. Примеры использования некоторых свойств компонентов: видимость, доступность и др.

Практическое занятие № 7. Создание проекта программы с использованием оператора вариантов. Компонент ListBox. Пример реализации программы.

Практическое занятие № 8. Составление программ с использованием оператора if и switch. Контрольная работа № 2 по теме "Разветвленные процессы".

Тематический план лекций

Лекция 1. **Архитектура персонального компьютера.** Понятие информации. Единицы количества информации и объема памяти. Представление информации в ЭВМ. Файлы, файловая система. Операционные системы для персональных компьютеров и их назначение. Операционная система Windows. Этапы решения вычислительных задач на компьютере.

Лекция 2. **Алгоритм, его свойства и способы описания.** Понятие программы. Типы данных С++: переменные, идентификатор, константы. Стандартные функции простых типов данных. Арифметические выражения. Структура программы-модуля проекта в С++ Builder. **Программы с линейной структурой.** Подпрограммы: описание, структура и параметры подпрограмм. Вызов подпрограмм. Примеры подпрограмм.

Лекция 3. **Разветвленные процессы.** Управление последовательностью выполнения операторов. Операции отношения и логические операции. Логические выражения и приоритет выполнения логических выражений. Операторы передачи управления.

Лекция 4. **Разветвленные процессы. Оператор вариантов switch.** Условные операторы. Примеры алгоритмов и проектов программ разветвленных процессов.

Дополнительная литература:

1) **Кузнецов В.Д., ЛеоненкоЛ. Л.** Архитектура и системное обеспечение персональных компьютеров. – Одесса: УДАЗ, 1997. – 58 с.
2) **Шаповаленко В.А. и др.** Информатика, ч. 1 / В.А. Шаповаленко, К.А. Богатко, В.Д. Кузнецов, И.Г. Швайко, И.А. Ещенко. – Одесса: ОНАС, 2003. – 102 с.
3) **Леонов Ю.Г.,** Угрик Л.М., Швайко И.Г. Сборник задач по программированию. – Одесса: УДАЗ, 1997. – 78 с.
4) **Леонов Ю.Г.,** Силкина Н.В., Шпинова Е.Д. Программирование инженерных задач: Метод. пособие с элементами лабораторного практикума. – Одесса: ОНАС, 2002. – 68 с.
5) **Майкл И., Хаймен. К. Borland C++.** – СПб.: Диалектика, 1995 – 416 с.
6) **Архангельский А.Я.** Програмирование в С++ Builder 5. – СПб.: Бином, 2000. – 1152 с.
7) **Березин Б.Н.**, Березин С.Б., Начальный курс С и С++. – М.: Диалог-МИФИ, 2000. – 288 с.
8) **Культин Н.Б.** С++ Builder в задачах и примерах. – СПб.: БХВ-Петербург, 2005. – 336 с.
9) **Страуструп Бьерн.** Язык программирования С++. – СПб.: Г.: Бином, 1999. – 991 с.

Конспект лекций

Лекция 1

Архитектура персонального компьютера. Понятие информации

1.1 Основные устройства компьютера и их свойства

Принципиальная конструкция современных компьютеров опирается на схему фон Неймана. Эта схема определяет функции отдельных частей компьютера (рис. 1.1).

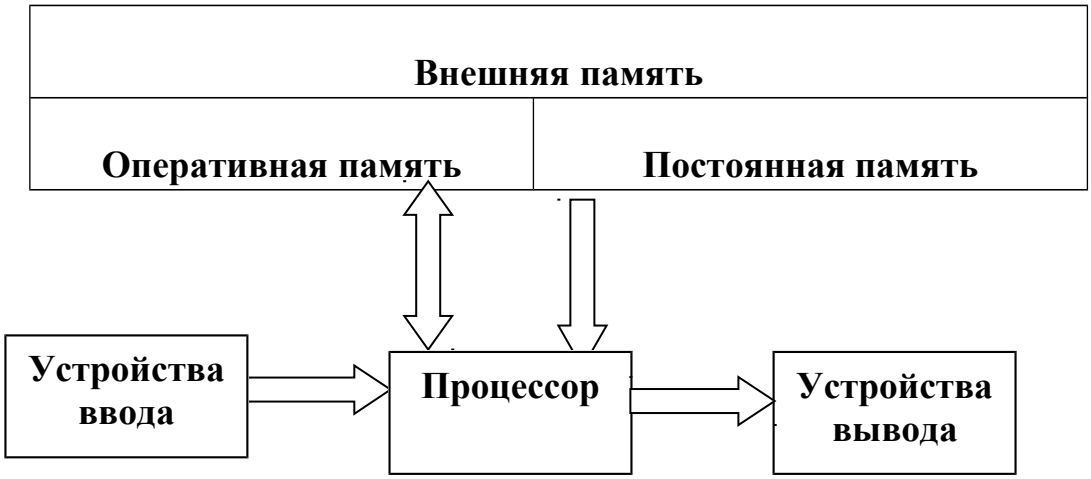


Рисунок 1.1– Схема фон Неймана

В соответствии со схемой фон Неймана, обработка информации выполняется процессором. Все действия процессора заданы программой — принцип программного управления. Данные и программы во время работы хранятся в оперативной памяти; для долгосрочного хранения те и другие переводятся из оперативной во внешнюю память. При этом пользователь вводит данные через устройства ввода (клавиатура, мышка, сканер, микрофон), а получает результат обработки через устройства вывода (монитор, принтер, акустические колонки).

Все программы и данные для работы процессора хранятся в памяти.

Объем устройств памяти определяется максимальным количеством информации, которое они могут хранить.

Оперативная память не может хранить данные при отсутствии электропитания. Для хранения данных без электропитания применяются разные виды внешней памяти. Самые распространенные в настоящее время устройства внешней памяти — дисковые, чаще всего магнитные диски.

Постоянная память содержит программы, с которых начинается работа ЭВМ. Без этих программ компьютер не сможет получить программы и данные из внешней памяти. Постоянная память не зависит от электропитания. Объем ее небольшой. Для изменения данных в ней нужно специальное устройство — программатор. В современных персональных компьютерах такое устройство

есть, но используют его редко — только тогда, когда нужно исправить ошибки в базовых программах.

Определяя назначение каждого устройства, схема не указывает способов изготовления, принципа работы устройств и методов связей между ними.

Принципы, которые определяют соединение устройств или их внутреннее оборудование, называют архитектурой. Почти все современные компьютеры сконструированы на основе шинной архитектуры. В соответствии с этой архитектурой, разные устройства связываются между собой с помощью общего канала — шины. В некоторых случаях шину называют магистралью. К общему каналу подключаются разные устройства (процессор, оперативная память и др.). С шиной они часто соединены специальными устройствами-посредниками — контроллерами и адаптерами. К одному контроллеру может присоединяться несколько устройств, которые работают по известным ему стандартам. Контроллеры бывают самые разнообразные — для подключения внешних устройств ввода-вывода, дисков и т.п.

Современные ЭВМ собираются из *ультрабольших интегральных схем (*УБИС). УБИС — это небольшие пластинки кремния, на которых вкраплениями других материалов созданы отдельные логические элементы. Такие элементы расположены очень плотно, что позволяет разместить на небольшой пластинке сложную схему.

Существует огромное количество микросхем для различных применений. Для решения конкретной задачи их часто выпускают целыми наборами — *чипсетами.*

Микросхемы собираются на платах — пластинах непроводящего материала, на которых закреплены проводники. К проводникам присоединяются микросхемы (припаиваются или вставляются в заранее припаянные разъемы).

Современные компьютеры конструируют, исходя из двух основных принципов: блочно-модульного и принципа открытой архитектуры.

Блочно-модульный принцип заключается в том, что отдельные по своим функциям устройства выполняются в виде отдельных модулей.

Принцип открытой архитектуры означает, что конструкторы ЭВМ предоставляют информацию о том, как разрабатывать устройства для нового компьютера.

Современные компьютеры собирают из отдельных частей, как конструктор, причем устройства пытаются выполнять в виде отдельных модулей, а их контроллеры - в виде плат.

Некоторые основные устройства (сама шина, основные контроллеры и др.) собирают в одном модуле — материнской плате. Другие устройства подключаются к материнской плате через специальные разъемы.

Если какое-то устройство выходит из строя, то весь компьютер, как правило, не ремонтируется, а заменяется только неисправный модуль. Если нужно подключить новое устройство, то можно разработать новый модуль-контроллер и встроить его в уже существующую ЭВМ.

Многие современные устройства и контроллеры — почти компьютеры. Они имеют свои процессоры, оперативную память, хранят и выполняют небольшие программы. Только это специализированные компьютеры, например, для вывода сложных изображений на экран.

1.2 Программное обеспечение и его типы

Для обработки информации компьютеру нужна последовательность команд – программа.

Программы необходимы для функционирования ЭВМ: без них компьютер не работает.

Каждый процессор имеет свой набор команд, т.е. множество операций, которые он "умеет" выполнять. Эти операции записываются в двоичном коде и для выполнения должны находиться в оперативной памяти.

Для первых ЭВМ все программы записывались в двоичном коде. Причем для решения каждой задачи такая программа создавалась и вводилась в память отдельно. После выполнения ее удаляли и вводили новую.

В настоящее время, когда память достаточно недорога и доступна, применяют другой подход. Программы для компьютеров не удаляют после выполнения, а накапливают.

Набор программ, разработанный для компьютера, называют *программным обеспечением.*

Программное обеспечение (ПЗ) можно разделить на классы: системное, прикладное, а также класс сред для разработки программ.

Системное программное обеспечение - программы, которые обеспечивают работу других программ. Они позволяют хранить библиотеку программ, находить нужные программы и запускать их на выполнение, а также распределять ресурсы между программами во время работы.

Прикладное программное обеспечение — программы, предназначенные для решения конкретных прикладных задач: редактирование текста или графики, выполнение определенных расчетов и т. п.

Среды для разработки программ — специальные программы, которые позволяют создавать новые системные и прикладные программы.

Важнейшая часть системного программного обеспечения собрана в комплексе программ, который называется *операционной системой* (ОС).

Операционная система выполняет такие функции:

- обеспечивает запуск программ;
- распределяет ресурсы компьютера между программами во время работы;
- предоставляет другим программам возможности работы с различными устройствами;
- предоставляет средства организации интерфейса пользователя.
- В состав современных операционных систем, как правило, входят несколько подсистем, основные из которых здесь перечислены:
- подсистема управления процессами;
- файловая подсистема;

- драйверы – специальные программы, которые стандартизируют работу с аппаратурой;
- функции для организации взаимодействия программ с пользователем;
- служба безопасности - разграничение прав доступа.

Самые распространенные на данное время операционные системы — системы Windows фирмы Microsoft.

Большинство компьютерных программ взаимодействует с пользователем.

Интерфейс — способ взаимодействия пользователя с компьютером, т.е. правила, по которым отдаются команды и демонстрируются результаты их выполнения. В последнее время чаще всего используется оконно-графический интерфейс, когда работа организуется с помощью окон, изображенных на экране.

Сегодня создано большое количество самых разнообразных программ. Благодаря этому разнообразию программ можно, как правило, самому не писать программу, чтобы решить определенную задачу, а воспользоваться уже готовым программным продуктом. Для решения часто встречающихся задач, разработаны специальные *информационные технологии,* т. е. такие способы и приемы, которые позволяют решать большие классы задач.

Информационные технологии существовали и до компьютеров (например, каталогизация библиотек), но именно компьютеры позволили автоматизировать обработку данных и связывать эти технологии между собой в единое целое.

1.3 Информация и ее представление. Системы счисления

Информатика — научное направление, которое занимается изучением законов, методов и способов накопления, обработки и передачи информации с помощью ЭВМ (электронно-вычислительных машин, или компьютеров) и других технических средств.

Информация - сведения об окружающем мире, что повышают уровень осведомленности человека. Пока информации было немного, люди могли получать и обрабатывать ее без посредников. Увеличение объема информации привело к необходимости ускорения ее обработки. Для этого были разработаны механизмы, автоматизирующие обработку информации. В настоящее время самым совершенным устройством переработки и хранения информации является компьютер. Для машинной обработки информацию нужно записывать, обозначая буквы числами, т.е. кодировать ее. Поэтому необходимо знать способы записи чисел.

Системой счисления называют правила записи чисел с помощью некоторого набора знаков. В зависимости от способа использования этих знаков системы счисления разделяются на *позиционные и непозиционные.*

В непозиционных системах счисления каждый знак обозначает всегда одно и то же число, и значения знаков в записи подытоживаются. Поэтому для записи больших чисел приходится вводить все новые и новые знаки.

Непозиционные системы неудобны для записи больших чисел и для выполнения арифметических действий.

Одна из непозиционных систем счисления используется и сегодня – *это римская система счисления.*

В римской системе счисления для небольших чисел используются такие знаки: I – один; V – пять; X – десять; L – пятьдесят; C – сто; D – пятьсот; M – тысяча.

В позиционных системах счисления один и тот же символ имеет разное количественное значение в зависимости от его позиции относительно других символов. Поэтому в позиционных системах для записи каких-нибудь чисел используется ограниченный набор знаков ― цифр.

Самым распространенным способом записи чисел в позиционной системе является десятичная система счисления. Каждое число записывается сочетанием десяти цифр, в котором вес конкретной цифры зависит от ее позиции ― разряда. Разряды отсчитываются справа налево. Первый разряд называется разрядом единиц, второй ― десятков, третий ― сотен и т.д. Число в десятичной системе счисления можно представить с помощью операций сложения, умножения и возведения в степень. Например:

$4321 = 4 \cdot 10^3 + 3 \cdot 10^2 + 2 \cdot 10^1 + 1 \cdot 10^0$.

Кроме десятичной системы счисления, существуют и другие позиционные системы: двоичная, троичная, четверичная, восьмеричная, шестнадцатеричная и т.д. Их названия отвечают основанию систем счисления. Основание системы счисления – количество цифр, допустимых в записи числа. Если число записано в позиционной системе счисления, отличной от десятичной, то основание указывается нижним индексом.

К примеру, если основание системы счисления более 10, то числа, превышающие 9, помечают последовательно буквами латинского алфавита. Например:

$$AD2F16 = 10 \cdot 16^3 + 13 \cdot 16^2 + 2 \cdot 16^1 + 15 \cdot 16^0 = 44335_{10}.$$

В компьютерах используется двоичная система счисления.

Поскольку запись числа в двоичной системе выходит достаточно длинной, с целью уменьшения его длины часто используют также восьмеричную или шестнадцатеричную системы счисления.

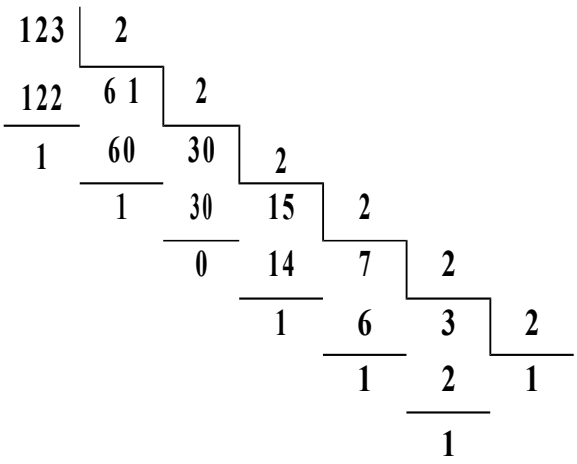
Для перевода числа из двоичной системы в десятичную достаточно записать его в виде суммы дополнений и подсчитать результат. Например:

$1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^2 + 1 \cdot 2^0 = 229_{10}$.

Аналогично осуществляется перевод из какой-либо другой позиционной системы счисления в десятичную.

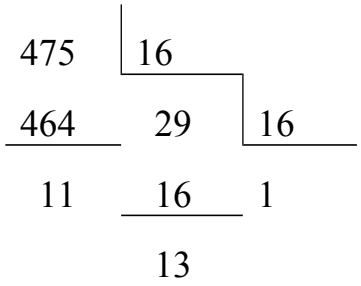
Перевод чисел из десятичной системы в систему с произвольным основанием заключается в нахождении остатков от деления числа на степень основания той системы, в которую нужно перевести число. Последовательность этих остатков и есть запись числа в новой системе. Разряды отсчитываются справа налево. Делить нужно до тех пор, пока не будет получен окончательный остаток.

Пример 1. Перевести из десятичной в двоичную систему счисления число 123.



$123_{10} = 1111011_2$

Пример 2. Перевести число 475 из десятичной системы в шестнадцатеричную.



$475_{10} = 1DB_{16}$

1.4 Единицы измерения информации

Поскольку в компьютерах используется запись информации в двоичной системе счисления, то количество информации измеряют, подсчитывая количество двоичных разрядов, необходимых для ее записи. Для удобства приняты такие единицы измерения информации:

1 бит – один символ, может хранить только значение 0 или 1;
1 байт = 8 бит;
1 Кбайт = 1024 байта;
1 Мбайт = 1024 Кбайта;
1 Гбайт = 1024 Мбайта.

Обратите внимание на то, что единицы измерения информации основываются на степенях числа 2. Десятичные префиксы (кило, мега и т. д.) дописываются только условно, поскольку $2^{10} = 1024$ – число, близкое к 1000.

1.5 Файловая система

Важнейшая для пользователя часть операционной системы — работа с внешней памятью, т.е. с хранилищем программ и данных.

Наибольшей логической единицей внешней памяти в системе Windows является том. Как правило, его по традиции называют *диском.* На больших носителях может быть не один, а несколько томов — *логических дисков,* т.е. том не всегда является физическим устройством. В операционных системах Windows каждый том отражается большой буквой латинского алфавита. Поэтому внутри информация организована в файловую систему.

Файловая система — способ организации хранения информации на носителях внешней памяти. Обеспечением работы с ней в операционной системе занимаются специальные компоненты.

Файл — область внешней памяти, обозначенная именем. Правила именования и выделения областей зависят от конкретной операционной системы.

В системе Windows имя файла состоит из двух частей, разделенных точкой: собственно имени файла и его расширения. Например, в имени файла **name.txt** имя - это **name**, а буквы **txt** означают, что файл является текстовым. Имена файлам рекомендуется давать, учитывая их содержание, так, чтобы по названию можно было понять, какую информацию содержит файл.

В операционных системах Windows в имени файла может насчитываться до 250 символов. Это могут быть символы русского и латинского алфавитов, цифры и некоторые знаки препинания. В именах файлов нельзя использовать символы «/», «\», «*», «:» «?», «"», «<», «>», «|»: они используются для записи команд.

Файлы отличаются между собой не только именами, но и содержанием. В зависимости от типа содержания файлам дают разные расширения. Некоторые расширения перечислены ниже.

Таблица 1.1 — **Расширение файлов**

Расширение	Содержание
EXE	Программа, готовая для выполнениня операционной системой
COM	Программа в старом формате
SYS	Часть операционной системы
DLL	Библиотека функций для различных программ
OAO	Команды для операционной системы
DOC	Документ
TXT	Текстовый файл
BMP GIF JPG	Изображение

Одно и то же содержание может быть записано в файлы с разным расширением, и файлы при этом будут отличаться *форматом.*

Формат файла — правила хранения информации в файле.

Большинство расширений — это сокращенные названия форматов. Жестких правил на названия расширений нет, и каждый разработчик может придумать собственный формат файла и расширения для него. Однако рекомендуется придерживаться общепринятых стандартов и не давать файлам собственного формата названия с известным расширением.

Каталогом называют специальный файл, в котором операционная система хранит информацию о других файлах (в частности о других каталогах). На каждом устройстве есть так называемый *корневой каталог* – основной каталог диска. Для большей ясности каталоги в операционной системе Windows называют *папками.*

Чтобы точно указать местонахождение файла, используется путь к файлу.

Путь к файлу — указание точного местонахождения файла. В нем слева направо последовательно указываются том (диск), на котором находится файл, и все папки, которые нужно раскрыть, чтобы добраться до файла. После диска ставится двоеточие, потом косая черточка. Папки разделяют косой черточкой. Например: C:\ Windows\Мои документы.

Полное имя файла включает путь в файл и имя файла.
Например, C:\Windows\Мои документы\workl.doc – полное имя файла work1.doc, который находится в папке «Мои документы» на диске С.

1.5 Основные операции с файлами, буфер обмена

Для работы с файлами существует несколько стандартных операций, которые поддерживают все операционные системы: копирование, перемещение, удаление.

Копирование — переписывание информации в новое место, причем при этом в старом месте информация сохраняется.

Перемещение — перенесение информации в новое место, после чего в старом месте она автоматически стирается.

Удаление — стирание записи о файле из каталога.

В операционных системах Windows операции копирования, перемещения и удаления часто выполняются с помощью программы «Проводник». Для выполнения копирования и перемещения используется буфер обмена.

Буфер обмена — специальное средство, предоставляемое операционной системой программам для обмена информацией.

Для работы с буфером обмена предусмотрены три команды:

вырезать — информация записывается в буфер обмена и удаляется со старого места;

копировать — копия информации записывается в буфер обмена;

вставить — информация из буфера обмена вставляется в новое место.

Таким образом, чтобы скопировать файл, нужно его найти, выделить, скопировать в буфер обмена, найти новое место и вставить его туда. Чтобы

переместить его, нужно файл не копировать, а вырезать. Этим способом копируют и перемещают не только файлы, но и другую информацию.

Для удаления файла в программе «Проводник» есть специальная команда, которая так и называется — «Проводник». Найти ее можно в пункте меню «Файл» или на инструментальной панели в программе «Проводник».

Для открытия файла на его ярлыке нужно выполнить двойной щелчок левой кнопкой мыши. Последующие действия будут зависеть от типа файла.

Если этот файл — программа (расширение EXE или COM), операционная система попробует ее запустить. Если это документ, по расширению или содержанию которого операционная система может определить его тип, она вызовет программу, которая и будет с этим файлом работать (например, запустит программу Microsoft Word).

Если же системе ничего о файле не известно, то она выдаст окно, в котором попросит выбрать программу для последующей работы.

Специальное назначение имеют файлы-ярлыки. В таком файле содержится ссылка на другой файл (каталог, программу, документ и т.п.). Двойной щелчок по нему откроет тот объект, на который ссылается ярлык.

Таким образом, операционные системы Windows упрощают работу пользователей, скрывая детали и показывая информацию в привычном для них виде.

1.6 Этапы решения вычислительных задач на компьютере

1. Математическое описание задачи (Постановка задачи).
2. Составление алгоритма решения задачи.
3. Создание проекта решения задачи.
4. Создание текста программы на алгоритмическом языке, по созданному алгоритму.
5. Ввод текста программы и создание визуального окна решения задачи.
6. Работа над ошибками в программе (отладка программы).
7. Выполнение программы и анализ полученных результатов.

Лекция 2

Алгоритм. Понятие программы. Программы с линейной структурой

2.1 Алгоритм, его свойства и способы описания

Алгоритм — точное распоряжение исполнителю (человеку или автомату) выполнить последовательность действий, направленных на достижение поставленной цели.

Алгоритм всегда составляется для конкретного исполнителя, т.е. для человека или автомата, который может его выполнить.

Совокупность всех команд, которые исполнитель может выполнить, и всех состояний объектов, которые он в состоянии распознать, называется *системой команд исполнителя.*

От обычных инструкций алгоритм отличается несколькими важными свойствами, которые допускают его автоматическое выполнение:

дискретность: алгоритм состоит из отдельных команд, которые выполняются последовательно, начало и конец их выполнения строго фиксированы;

понятность: команды алгоритма должны быть полностью понятными исполнителю;

точность: после выполнения каждой команды точно известно, завершено ли выполнение алгоритма или нужно перейти к следующей команде;

результативность: алгоритм завершается либо достижением цели, либо выявлением невозможности решения задачи;

массовость: алгоритм применяется к какой-нибудь корректной формулировке задачи, для решения которой он разработан.

Каждый алгоритм разрабатывается в строго предопределенных условиях, которые содержат четкую формулировку задачи и систему команд исполнителя.

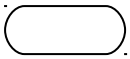
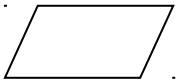
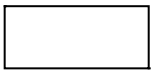
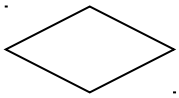
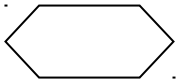
Для того чтобы алгоритм мог выполняться автоматом, его нужно записать в той форме, в которой автомат его сможет "читать". Для ЭВМ такой формой является двоичный код.

Однако человеку записывать алгоритм в машинном коде крайне неудобно, поскольку на запись даже простого алгоритма тратится много времени, и появляется большое количество ошибок. Поэтому для составления и подготовки программ используются другие формы записи алгоритма, в частности блок-схемы и языки программирования.

Блок-схема — последовательность, составленная из отдельных соединенных между собой в порядке выполнения блоков. С помощью блок-схем описывают структуру программы. Вид блоков определяется их назначением в программе. Форма блока определяет вид действий, а записи внутри - подробности (параметры).

Конфигурация блоков представлена в таблице 2.1.

Таблица 2.1 **– Конфигурация блоков**

Название блока	Вид блока	Назначение в программе
Начало (конец) или вход (выход)		Начало (конец) или вход (выход) для подпрограмм
Блок ввода (вывода)		Ввод данных и вывод результатов вычисления
Блок вычислений		Вычислительный процесс (вычисление по формулам)
Логический блок (разветвление)		Выбор направления выполнения алгоритма, в зависимости от условий
Блой модификаций (заголовок цикла)		Автоматическое изменение переменной
Ссылка на вторую страницу		Указывает связи между частями схемы, расположенными на разных страницах

2.2 Язык программирования

Язык программирования **–** формализованный язык, предназначенный для описания алгоритмов решения задач на ЭВМ.

Языки программирования бывают низкого, среднего и высокого уровня.

Язык программирования низкого уровня **–** язык программирования, структура команд которого определяется системой команд процессора и архитектурой ЭВМ. Часто эти языки называют *языками ассемблера.*

Писать программы языками ассемблера достаточно просто, поскольку, в сущности, эти вещи являются просто буквенными записями машинных команд.

На основе языков ассемблера были созданы *языки среднего уровня* или *языки макро-ассемблера.* В них система команд расширена более сложными конструкциями. Специальные программы-переводчики сами переводят эти конструкции в двоичный код.

Языки высокого уровня **–** языки программирования, средства которых допускают описание алгоритма в наглядном виде, т.е. не на основе команд процессора, а на основе слов естественного языка.

Программа на таком языке переводится на машинную с помощью *программы-транслятора,* которая переводит конструкции языка программирования на язык команд процессора.

Существует два вида трансляторов: интерпретаторы и компиляторы.

Интерпретаторы **–** трансляторы, которые выполняют команды сразу после их получения.

Компиляторы – трансляторы, которые сначала переводят всю программу в машинный код, а затем ее можно запустить на выполнение.

Программа в режиме интерпретации работает медленнее, однако можно отслеживать выполнение программы и при этом даже изменять ее. К тому же для того, чтобы выполнить программу, нужно всегда запускать программу-транслятор, что требует больше дополнительных ресурсов.

Алфавит языка – набор символов, которые можно использовать для записи команд в программе, это большие и малые латинские буквы, цифры и специальные знаки.

Переменная – именуемый участок оперативной памяти. Применяется для хранения данных во время работы программы.

Значение переменной – фактическое значение, которое находится в этой области памяти.

Тип переменной – вид данных, которые переменная может хранить. Переменные определенного типа хранятся и обрабатываются по определенным правилам. Тип переменной также указывает, какой объем памяти займет переменная.

2.3 Язык программирования С++. Типы данных языка С++

В программе, написанной на языке С++, все переменные должны быть определены, т.е. для каждой переменной должен быть указан ее тип.

Данные целого типа определяют множество целых чисел в разных диапазонах. Существует несколько целых типов, которые отличаются допустимым диапазоном значений и размерами оперативной памяти, которую они занимают. Целый тип отражается идентификаторами int, unsigned int, характеристики которых приведены в таблице 2.2.

Таблица 2.2 – **Диапазон значений типов данных**

С++	Размер (в байтах)	Диапазон	Тип данных
char	1	–128…126	символьный
unsigned char	1	0…255	
short	2	–32 768…32 767	короткий
unsigned short	2	0…65 535	
int	4	–2 147 483 648…2 147 483 647	целый
unsigned int	4	0…4 294 967 295	
float	4	3.4·10-38…3.4·1038	действительный
double	8	1.7·10-308…1.7·10308	действительный
long double	10	3.4·10-4932…1.1·104932	длинный
bool	1	true, false	логический

Данные действительного типа – это данные, которые в качестве значений принимают числа с фиксированой точкой или плавающей точкой.

Данные действительного типа обозначаются идентификаторами float и double и имеют характеристики, приведенные в таблице 2.2.

Константы символьного типа **—** это какой-нибудь символ языка, взятый в апострофы, например: '–' , 'A', '7'.

Переменная символьного типа **(**char) – это переменная, которая принимает значение символьной константы.

Все типы языка С++ разделяют на две группы: основные типы и структурированные.

К основным типам можно отнести char, int, float и double, а также их варианты с модификаторами short (короткий), long (длинный), signed (со знаком) и unsigned (без знака).

Например:

```
int a,b,c;
short t;
unsigned int ui;
a=17;
b=-197;
t=-32767;
double g;
g=-0.123E-3.
```

Объявлять переменные можно в любом месте программы, но перед их использованием.

Структурированные типы включают в себя указатели, массивы разных типов, типы функций, классы, структуры.

2.4 Константы в языке С++

Константа – величина, которая не изменяется в процессе выполнения программы. Константы могут быть разного типа, например, некоторые из них представлены в таблице 2.3.

Таблица 2.3 **— Типы констант**

Тип данных	Константа
char	'a', '\n', '9'
int	1, 153, -349
float	123.23, 6.34E-3, 4E+5
double	23.23, 12312312, -0.947

В языке С++ используются строковые константы. Строковые константы, или строка представляют собой набор символов в двойных кавычках, например ″Программирование″, ″Академия связи″.

Для объявления константы в программе используется модификатор const. При этом указывается тип константы. Например:

const float Pi=3.14159;

В качестве значения константы можно указывать константное выражение, которое содержит объявление константы. Например:

const float Pi2=2*Pi;

const float K=Pi/180.

Для целых констант тип можно не указывать.

2.5 Операции языка C++

Операции сложения, вычитания, умножения и деления, в языке C++, выполняются слева направо, т.е. сначала вычисляется значение левого операнда, а затем — значения правого. Если операнды имеют один и тот же тип, то результат тоже будет иметь тот же тип. Поэтому, если операция деления применяется к целым переменным, то результат будет тоже целым.

Арифметические операции и их запись в языке C++ представлены в таблице 2.4.

Таблица 2.4 —**Арифметические операции**

Обозначение	Операция	Пример
+	сложение	X+Y
-	вычитание и унарный минус	X-Y
*	произведение	X*Y
/	деление	X / B
%	остаток от целочисленного деления	I % 6
++	увеличение на единицу (инкремент)	i++ ++i
--	уменьшение на единицу (декремент)	i-- --i

В языке C++ используются еще две операции, которых нет в других языках. Это унарные операции ++ (инкремент) и -- (декремент). Унарной называется операция, которая имеет только один операнд. Операция ++ добавляет единицу к операнду, операция -- вычитает единицу из операнда. Обе операции могут стоять перед операндом (префиксная форма записи) и после операнда (постфиксная форма записи). При префиксной форме переменная сначала увеличивается или уменьшается на единицу, а затем ее новое значение используется в том выражении, в котором она встретилась. При постфиксной форме в выражении используется текущее значение переменной, и только затем переменная увеличивается или уменьшается на единицу. Три написанных ниже оператора дают один и тот же результат, но различаются при использовании в выражениях:

i = i+1; ++i; i++;

Например, в результате выполнения операторов:

```
int  i = 1,  j;
j = i++ * i++;
```

значение переменной **i** будет будет равняться 3, а переменной **j** = 1.

Если изменить эти операторы таким образом:

```
int  i = 1,  j;
j = ++ i * ++i;
```

то результат будет иным: значение **i** будет равняться 3, а значение **j**= 9.

Оператор присваивания – основной оператор языка программирования – имеет такой формат:

имя переменной = выражение;

Этот оператор вычисляет значение выражения и присваивает вычисленное значение переменной; при этом тип выражения должен соответствовать типу переменной. Допускают присваивание переменной вещественного типа значения выражения целого типа.

Присваивание переменной целого типа выражения действительного типа **запрещено!**

Кроме простого присвоения все остальные являются сложными операциями. Они присваивают результат переменной слева, указанной перед символом «=». Например, выражение X+=Y эквивалентно выражению X=X+Y, но записывается более компактно и выполняется быстрее.

2.6 Стандартные функции преобразования типов данных

Ввод и вывод данных в C++ происходит с помощью переменных строкового типа. Если написать, например в Edit1, 3.25, то это не число, а его изображение в виде строки символов. С ним никаких вычислений делать нельзя; нужно преобразовать тип string на числовой тип, а затем наоборот. В языке C++ это осуществляют стандартные подпрограммы. Приведем примеры их применения:

```
X=StrToFloat(S);     //функция, которая преобразует string-строку  S в вещественное число X;
X=StrToInt(S);    // функция, которая преобразует string-строку  S в целое число X;
S=FloatToStr(X);  // функция, которая преобразует вещественное  число X в string-строку  S;
S=IntToStr(X);  //функция, которая преобразует целое  число X в string-строку S.
```

2.7 Основные математические функции

Описание математических функций и запись функций на языке программирования C++, представлено в таблице 2.5.

Таблица 2.5 – **Математические функции**

Функция	Описание	Библиотека
int **abs**(int i)	модуль целого числа **x**	stdlib.h
double **ceil**(double x)	округление вверх: наименьшее целое, не меньше чем **x**	math.h
double **exp**(double x)	экспонента	math.h
double **fabs**(double x)	модуль вещественного числа **x**	math.h
double **floor**(double x)	округление вниз: наибольшее целое, не большее чем **x**	math.h
Extended **IntPower**(Extended Base, int Exponent)	возвести **Base** в целую степень **Exponent**	Math.hpp
double **log**(double x)	натуральный логарифм	math.h
double **log10** (double x)	десятичный логарифм	math.h
Extended **LogN**(Extended Base, Extended X)	логарифм **X** по основаннию **Base**	Math.hpp
double **pow**(double **x**, double **y**)	$\mathbf{x}^{\mathbf{y}}$	math.h
double **sqrt**(double x)	корень квадратный	math.h
double **acos**(double x)	арккосинус	math.h
double **asin**(double x)	арксинус	math.h
double **atan**(double x)	арктангенс	math.h
double **cos**(double x)	косинус	math.h
double **sin**(double x)	синус	math.h
double **tan**(double x)	тангенс	math.h

При работе с математическими функциями надо иметь в виду, что файлы math.h и Math.hpp в C++ Builder автоматически не подключаются к модулю вашего приложения. Поэтому для использования описанных в этих файлах функций необходимо вручную вводить директивы:

#include <math.h> // *подключение библиотеки math.h*

#include <Math.hpp> // *подключение библиотеки Math.hpp*

Арифметические выражения строятся из арифметических констант, переменных, функций и операций над ними. Вычисления выполняются слева направо в соответствии с таким старшинством операций:

1. Стандартные функции, ++, --.
2. Умножение (*), деление (/), остаток от деления (%).
3. Сложение (+) и вычитание (–).

Выражения в круглых скобках выполняются в первую очередь.

2.8 Среда программирования C++ Builder

C++ Builder – это технология визуального программирования, где автоматизирована ее трудоемкая часть – создание интерфейсных программ с диалоговыми окнами. Оболочка C++ Builder дает возможность вместо полного самостоятельного написания программы использовать большой набор готовых визуальных объектов, так называемых *компонентов*, а их пиктограммы размещены на соответствующих вкладках палитры компонентов. Компоненты обеспечены набором определенных свойств (название, цвет, размер и т.п.),

методов (программных кодов, готовых к выполнению) и шаблонов подпрограмм-отзывов на обработку событий (подпрограмм-обработчиков). Эти события (нажатие клавиши мыши, выбор пункта меню и тому подобное) возникают во время выполнения программы. Наиболее распространенным является событие OnClick – нажатие левой кнопки мыши на объекте.

Кроме того, C++ Builder позволяет работать с разными базами данных, создавать программы для работы в Интернет.

Свойства компонентов выбираются на странице *Properties*, а подпрограммы-обработчики – на странице *Events* в окне Object Inspector. Например, свойство AutoSize (Автомасштаб) имеет значение True (истина) без дополнительных указаний. Это означает, что размер компонента подстраивается под размер его содержания. Но можно задать необходимый размер с помощью свойств Width (длина) и Height (высота), если при этом изменить значение свойства AutoSize на False (ошибочность).

“Контейнером”, в котором на экране размещаются компоненты, служит форма, которая сама является компонентом с названием Form. Без дополнительных указаний заглавие компонента совпадает с его названием, к которой добавляется порядковый номер, начиная с 1. Но название можно изменить с помощью свойства Caption . Компоненты, которые обеспечивают интерфейс между программой и пользователем, называют базовыми. К базовым компонентам можно отнести компоненты, которые содержатся на вкладке **Standard** палитры компонентов. Перечислим некоторые из них:

 – Label, надпись, которая служит для написания объяснительного текста на форме с использованием свойств Caption и Font;

 – Button, кнопка предназначена для отклика на событие;

 – Edit, однострочный текстовый редактор, который служит для ввода и вывода лишь одной строки текста; эта строка хранится в свойстве Text.

Как только на форме появляется компонент, окно Object Inspector отображает информацию о нем.

Окно C++ Builder представлено на рис. 2.1.

В верхней части окна находится главное меню. Ниже расположены две инструментальные панели, которые содержат ряд кнопок и палитру компонентов. В основном поле окна слева расположен Инспектор Объектов, Form1 – окно пустой формы. Под формой расположено окно Редактора Кодов.

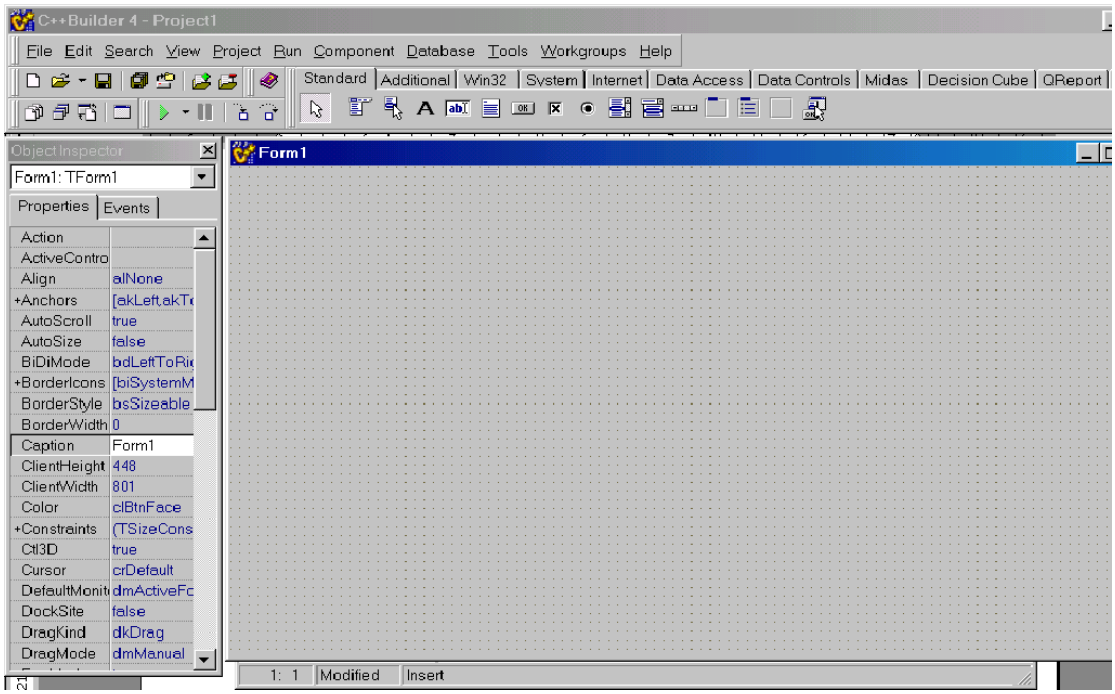


Рисунок 2.1 – Главное окно C++ Builder

2.9 Последовательность работы и организация проекта в C++ Builder

Процесс создания программы в **C++** Builder состоит из двух шагов: сначала нужно создать форму программы, а затем функции обработки событий. Последовательность работы в **C++** Builder может быть такой:

- вынести на форму с палитры компонентов все необходимые компоненты для создания диалогового окна программы;
- задать на вкладке **Properties** (свойства) Инспектора Объектов все необходимые свойства для компонентов, расположенных на форме;
- создать необходимые отзывы на события для расположенных на форме компонентов. Это можно сделать с помощью вкладки **Events** (события) Инспектора Объектов или непосредственно с формы, щелкнув дважды левой кнопкой мыши по компоненту;
- в окне Редактора Кодов заполнить отклики на события кодом, а также написать текст всех необходимых программ. Переключить окно формы и окно редактора можно клавишей **<F12>**;
- сохранить созданный проект с помощью команды главного меню **File/Save Project As**;
- запустить проект на выполнение одним из способов: нажать клавишу **<F9>**, выполнить команду меню **Run/Run**, нажать соответствующую кнопку на панели инструментов;
- если есть ошибки, курсор на них указывает по очереди и внизу экрана подается сообщение о них. Например:

statement missing; – ожидается точка с запятой(;);

undefined symbol “ X ” – неопределенный символ;
call to undefined function ‘cos’ – вызов неопределенной функции (т.е. не подключенна библиотека математических функций);
compound statement missing } – отсутствие операторной скобки }.

- если ошибок нет, появится форма без координатной сетки, если ошибки есть нужно их исправить и опять запустить проект на вычисление. Нужно дать команду на сохранение проекта без ошибок (SaveAll);
- провести расчеты, выполняя необходимые действия (вводить данные, нажимать нужные кнопки, и т. п.);
- выйти из C++ Builder. Сеанс закончен.

Организация проекта в C++ Builder: проект C++ Builder состоит из нескольких файлов: файлов форм, модулей, ресурсов, и т.п. Главной частью проекта в C++ Builder является головной файл проекта **(.cpp**) с функцией **WinMain**, с которой начинается выполнение программы и которая обеспечивает инициализацию других модулей. Он создается автоматически и не нужно его изменять; этот файл имеет имя **Project1**.

Основной файл, с которым вы работаете, – файл реализации модуля (**.cpp**). В нем хранится код, соответствующий форме. В заголовочном файле с расширением **.h** хранится объявление класса этой формы. Весь основной текст данного файла формируется автоматически, но иногда нужно вводить в него объявление своих функций, типов, переменных. Открыть этот файл в редактор кодов можно, щелкнув в окне с файлом реализации модуля правой кнопкой мыши и выбрав из локального меню команду **Open Sourse/ Header File.**

Имена заголовочного файла и файла реализации совпадают. Вы задаете это имя, когда в первый раз сохраняете проект. По умолчанию C++ Builder предлагает имя **Unit1.**

2.10 Программы с линейной структурой

Структура программы: программа на C++ состоит из объявления (переменных, констант, типов, классов, функций) и объявления функций. Среди функций всегда есть главная – **main** для консольных приложений (которая работает с WIN32) или **WinMain** для приложений Windows. Эта главная функция выполняется после начала работы программы. Функцию **WinMain** содержит главный файл проекта, который создается автоматически, и его изменяют только в исключительных случаях. Чтобы увидеть текст главного файла, нужно выполнить команду **Project/View Source.**

Программы на C++ создаются по модульному принципу и состоят из нескольких модулей. Все объекты компонентов располагаются на формах. Для каждой формы C++ Builder создает отдельный модуль.

Заголовочний файл имеет вид:

```
//---------------------------------------------------------------------------
#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <Classes.hpp>
```

```
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
// здесь могут размещаться дополнительные директивы
// препроцессора (в частности include),
// которые не включаются в файл автоматически
// объявление класса формы TForm1
class TForm1 : public TForm
{
__published:        // IDE-managed Components
// расположеные на форме компоненты
        TLabel *Label1;
        TButton *Button1;
        void __fastcall Button1Click(TObject *Sender);
private:        // User declarations
// закрыт раздел класса
// здесь могут размещаться объявления типов, переменных, функций,
// которые входят в класс формы, но недоступны для других модулей
public:                 // User declarations
// открытый  раздел класса
// здесь могут размещаться объявления типов, переменных, функций,
// которые входят в класс формы и  доступны для других модулей

__fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
// здесь могут размещаться объявления типов, переменных, функций,
// которые не включаются в класс формы;
// доступ к ним  из других блоков  только при соблюдении
// некоторых дополнительных условий
#endif
```

Начинается заголовочный файл строками, первый символ которых – #. С этого символа начинаются директивы препроцессора. Среди них наиболее важны директивы include. Эти директивы подключают к заданному модулю указанные в них файлы. В частности в заголовочном файле размещаются директивы include, подключающие копии файлов, содержащие описание компонентов, переменных, функций, используемых в данном модуле. Но для некоторых функций такое автоматическое подключение не делается. В этих случаях разработчик программ должен добавить соответствующие директивы include собственноручно. Например, если вы хотите в своей программе использовать математические функции, необходимо подключить библиотеку, которая содержит эти функции, т. е. собственноручно ввести директивы:

```
#include <math.h>     // подключение библиотеки math.h
```

#include <Math.hpp> // *подключение библиотеки Math.hpp*

После директив препроцессору идет описание класса формы – TForm1. Раздел __published заполняется автоматически в процессе проектирования формы. В этом примере вы видите объявление указаний на два компонента: Label1 типа TLabel и Button1 типу TButton. Там есть объявление функции Button1Click – обработчика события на нажатие кнопки Button1. Разделы private и public заполняются разработчиком программ. То, что объявлено в разделе public, будет доступно для других классов и модулей. То, что объявлено в разделе private, доступно только в пределах данного модуля.

Ниже раздела PACKAGE можно разместить объявление типов, переменных, функций, к которым будет доступ из других модулей, но которые не включаются в класс формы.

Теперь рассмотрим текст файла реализации.

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
// здесь могут размещаться дополнительные директивы
// препроцессора (в частности include),
// которые не включаются в файл автоматически
// объявление класса формы TForm1
TForm1 *Form1;
//---------------------------------------------------------------------------
// вызов конструктора формы Form1
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
// здесь можно разместить операторы,
// которые должны выполняться при  создании формы
}
//---------------------------------------------------------------------------
// здесь могут размещаться объявления типов, переменных,
// доступ к которым из других модулей возможен только при
// выполнении некоторых дополнительных условий;
// здесь должны быть  реализации  всех функций, которые объявлены  в
// заголовочном файле, а также  могут быть реализации  любых
// дополнительных функций, не объявленных ранее

void __fastcall TForm1::Button1Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------------
```

Пример создания проекта программы с линейной структурой:

Задача: Вычислить полную поверхность и объем прямого конуса с радиусом R = 12,54 см и образующей L = 24,88 см, по формулам:

$$S = \pi R^2 + \pi RL;\ V = \pi R^2 H/3,$$

где H – высота конуса, $H = \sqrt{L^2 - R^2}$. Значение R и L ввести с экрана.

Последовательность решения задачи:

Форма для данного проекта может быть, например такой, как на рис. 2.2.

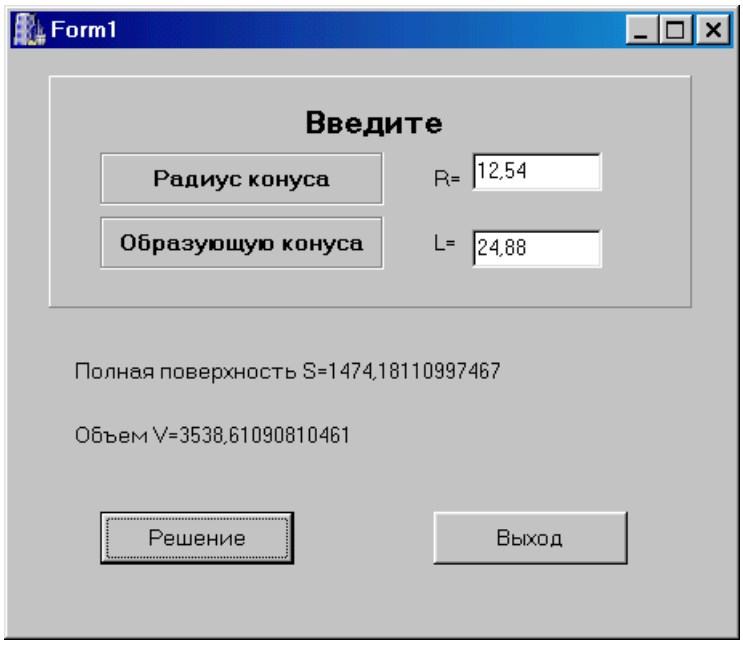


Рисунок 2.2 – Форма проекта вычисления полной поверхности та объема конуса

Текст файла реализации приведен ниже. Заголовочный файл для этого проекта создается автоматически и включает в себя описание всех компонент формы. Текст его не приводится.

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
#include <math.h>      // подключение библиотеки математических функций
TForm1 *Form1;
//---------------------------------------------------------------------------

__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{ Label4->Visible=0;          // сделать  невидимыми метки Label4
```

```
  Label5->Visible=0;          //          и  Label5
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
double R,L;
 R=StrToFloat(Edit1->Text);
 L=StrToFloat(Edit2->Text);
double S,V,H;
const float Pi=3.14159;
 S=Pi*R*(R+L);
 H=sqrt(pow(L,2) -pow(R,2));
 V=Pi*R*R*H/3;
 Label4->Visible=1; // сделать видимой метку Label4
 Label5->Visible=1; // сделать видимой метку Label5
 Label4->Caption="Полная поверхность S="+FloatToStr(S);
 Label5->Caption="Объем V="+FloatToStr(V);
 }
```

В конструкторе форм записаны операторы, которые делают невидимыми метки Label4 и Label5 в момент создания формы. Эти метки используются для вывода результатов на экран. Основные вычисления осуществляются в подпрограмме Button1Click. Обратите внимание, что после заглавия подпрограммы следует фигурная скобка – {. Фигурные скобки ({}) в C++ всегда используются парами (открывающая и закрывающая), их называют операторными скобками.

Строка double *R,L*;
объявляет действительные переменные R и L. Все переменные в языке C++ должны быть объявлены перед их использованием.

Строка const float Pi=3.14159;
объявляет действительную константу и присваивает ей значение.

Строка R=StrToFloat(Edit1->Text);
является оператором присвоения. Переменной R присваивается содержание редактора Edit1. Для преобразования значения из строкового типа в число действительного типа используется функция StrToFloat. При создании программы нужно помнить, что большие и маленькие буквы в C++ отличаются, т.е., например, переменные R и r являются разными.

Лекция 3
Разветвленные процессы. Операторы передачи управления

Разветвленным называется вычислительный процесс, если, в зависимости от определенных условий, он реализовывается по одной из определенных, предварительно предусмотренных, веток алгоритма.

Для программной реализации таких вычислений необходимо использовать операторы передачи управления, которые позволяют изменять порядок выполнения операторов программы. В языке С++ для этого предусмотрены операторы: безусловного перехода – **goto**, условного перехода – **if** и выбора варианта – **switch**. Для записи условия перехода необходимо использовать логические (булевые) выражения.

Оператор безусловного перехода goto **(**идти к) позволяет передать управление в какую-нибудь точку кода (программы), которая обозначена специальной меткой.

Оператор goto имеет вид:

goto <метка >;

Метки, на которые можно передавать управление, в языке С++ не объявляют. Каждая метка может отражаться допустимым в алгоритмическом языке идентификатором, который обязательно начинается с латинской буквы. Например:

start, M55, second, a;

означает четыре метки: start, M55, second, a.

В С++ оператор goto очень редко применяется.

Операции отношения и логические операции. В основе всех действий с информацией лежат так называемые *логические операции.*

Логические переменные – переменные, которые могут получать только два значения: ИСТИНА или ЛОЖЬ. Часто эти значения помечают цифрами 1 и 0 (1 – ИСТИНА, 0 – ЛОЖЬ).

Логическая операция – действие, которое выполняется над логическими переменными, ее результат также либо ИСТИНА, либо ЛОЖЬ.

Базовые логические операции: логическое добавление (операция ИЛИ), логическое умножение (операция И) и отрицание (операция НЕ).

Логическое сложение (ИЛИ) – логическая операция, результатом выполнения которой является значение ИСТИНА, если хотя бы одна из логических переменных имеет значение ИСТИНА. Записывается с помощью знака « || »: *A* || *B*.

Логическое умножение (И) – логическая операция, результатом выполнения которой является ИСТИНА, если все логические переменные имеют значение ИСТИНА, в остальных случаях результат – ЛОЖЬ. Записывается с помощью знаков **&&.**

Логическое отрицание (НЕ) – логическая операция, которая выполняется над одной логической переменной, ее результатом является значение ИСТИНА,

если начальным значением было ЛОЖЬ, и ЛОЖЬ – если было ИСТИНА. Записывается с помощью знака **!**.

Из логических переменных с помощью логических операций и скобок (для указания порядка действий) строятся *логические выражения.*

Логические отношения:

= = (равно), != (не равно), <, >, <=, >=

используются при сравнении двух выражений. Результатом такого сравнения могут быть значения **true** (истина) или **false** (ложь). Однако результатом логического выражения может быть и целочисленное арифметическое значение. При этом значение **0** расценивается как **false**, а любое ненулевое значение – как **true**. Рассмотрим пример:

```
int   tr, fal;
tr = (105<=109);
fal= (109>105).
```

В результате выполнения этих операторов переменная tr получит значение 1, а переменная fal будет равняться 0.

Ниже приведены операции, расположенные в порядке уменьшения их приоритетов.

1. ( ), ++, - -, ~ !
2. * / %
3. + –
4. >> <<
5. < <= > >=
6. == !=
7. &&
8. ||

Оператор if, имеет две формы: сокращенную и полную. Сокращенная форма имеет вид:

if (логическое выражение**)** оператор**;**

Полная форма этого оператора такова:

if (логическое выражение**)** оператор1;
else оператор2;

где if (если), else (иначе) – служебные слова; оператор1, оператор2 – простые или составные операторы языка.

Порядок выполнения условного оператора показан в виде структурной схемы алгоритма, представленной на рис. 3.1.

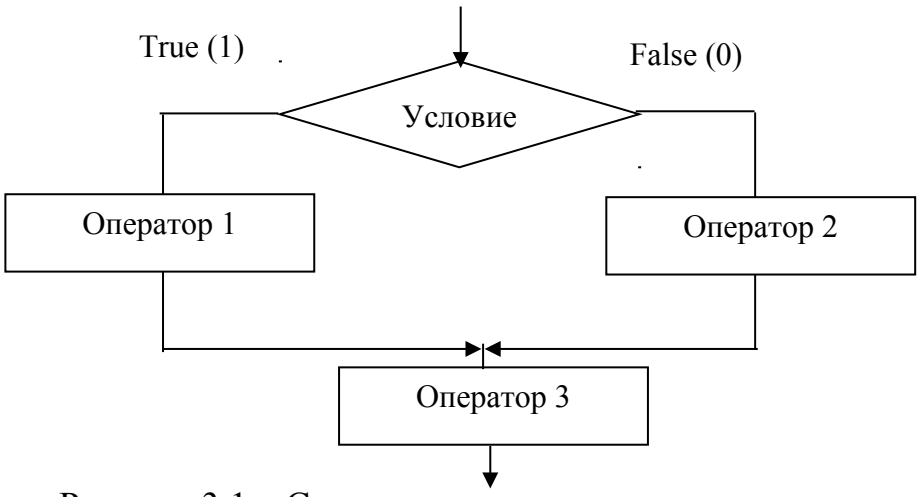


Рисунок 3.1 – Схема алгоритма условного оператора перехода

Как видно из схемы, если значение логического выражения равняется true, то выполняется оператор1, если логическое выражение – false, то выполняется оператор2 (оператор 1 пропускается). Затем, при любом исходе, выполняется оператор 3, который стоит за оператором if. Например, фрагмент программы вычисления функции

$$y = \begin{cases} \ln x, & \text{если } x > 0; \\ e^x, & \text{если } x \le 0 \end{cases}$$

имеет вид : if (x>0) y=ln(x); else y:=exp(x);

Здесь логическое выражение – отношение x>0, оператор 1 и оператор2 – операторы присвоения y=ln(x) и y=exp(x).

Порядок выполнения оператора краткой конструкции if (без else) показан на рис. 3.2.

Если логическое выражение принимает значение False, то выполняется следующий за if оператор.

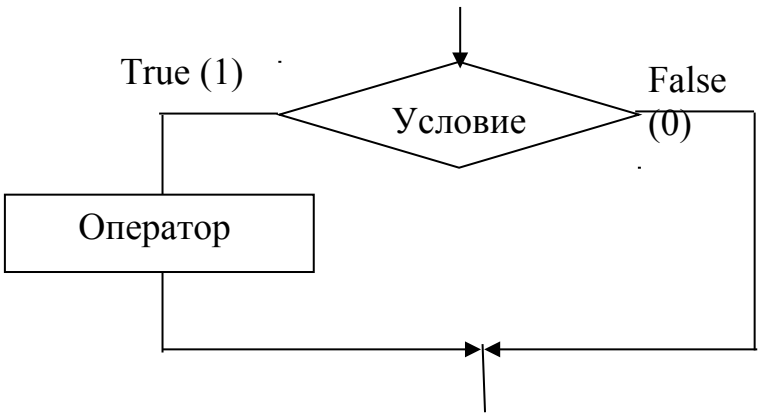


Рисунок 3.2 – Схема алгоритма краткого оператора условного перехода

Условные операторы могут быть неограниченно вложенными друг в друга.

В некоторых случаях сложно разобраться, в какой последовательности выполняются такие вложенные операторы. Очень часто встречается использование конструкции if – else – if. Например:

```
if (a>b)  c=5;
else  if (a>e)  c:=1;
else c:=2;
```

В такой конструкции условия операторов if проверяются сверху вниз. Как только какое-либо из условий принимает значение Истина, выполняется оператор, следующий за этим условием, а вся оставшаяся часть конструкции будет проигнорирована.

Нужно пытаться записывать подобные вложенные операторы максимально наглядно, используя отступы из пробелов, избегать запутанных составляющих и большого уровня вложенности.

Вывод сообщения реализуется функцией ShowMessage. Функция ShowMessage - предназначена для вывода на экран диалогового окна с текстом сообщения с одной командной кнопкой ОК. Общая форма записи функции такова: ShowMessage ("Сообщение");.

Пример: Ввести значения координат *(x, y)* точки плоскости. Определить, какой четверти плоскости принадлежит эта точка, если точка лежит на оси, то вывести сообщение "Точка лежит на оси".

Текст программы для решения этой задачи приведен ниже:

```
……
 void __fastcall TForm1::Button1Click(TObject *Sender)
   {
       float x=StrToFloat(Edit1->Text);
       float y=StrToFloat(Edit2->Text);
        int nomer;
            if((x = = 0)||(y = = 0))
               { ShowMessage("Точка лежит на оси");
                } else

     if ((x>0)&&(y>0))  nomer=1;        //точка принадлежит первой четверти
     else  if ((x<0)&&(y>0)) nomer=2; //точка принадлежит второй четверти
     else  if ((x<0)&&(y<0))  nomer=3;//точка принадлежит третьей четверти
              else  nomer=4;                // точка принадлежит четвертой четверти
       //вывод результатата решения задачи в компонентуEdit3
     Edit3->Text=IntToStr(nomer(x,y));
   }
```

Лекция 4

Разветвленные процессы. Оператор вариантов switch

Оператор switch является оператором выбора вариантов. Общая форма записи оператора:

```
switch  (выражение)
{
case значення_1:  {последовательность операторов; break;}

…
case значення_n:  {последовательность операторов; break;}
.
default:   последовательность операторов;
}
```

Сначала вычисляется выражение в скобках. Выражение должно иметь порядковый тип, например, целый, символьный и т. д. Значение выражения сравнивается со значениями меток. Значения, которые указаны в метках case**,** должны быть константными выражениями, соответствующими возможным значениям выражения выбора. Если значение выражения совпало со значением какой-либо метки, то выполняется последовательность операторов, обозначенная этой меткой и записанная после двоеточия. Если значение выражения не совпало ни с одной из меток, то выполняются операторы, следующие за меткой default. Метка default необязательно должна включаться в структуру switch.

Оператор break приводит к выходу из структуры switch и переходу к следующему оператору программы. Если оператор break отсутствует, то будут выполнены все операторы, начиная с операторов совпадающей метки к первому break или до конца программы.

Пример 1. Вычислить значение Y для трех вариантов параметров:

1) $a = -3{,}7$; $b = 5{,}6$; $c = \mathrm{tg}|bx|$,

2) $a = 0{,}81$; $b = -2{,}4$; $c = \mathrm{tg}|bx|$,

3) $a = 2{,}5$; $b = 0{,}6$; $c = \mathrm{tg}|bx|$

по формулам

$$Y = \begin{cases} \sin(e^{a+c}) + x^2, & \text{åñëè } x<a; \\ \sqrt[3]{a+|5{,}3b|} + c, & \text{åñëè } a \le x < b; \\ \cos^2 a + \sin x^2 - bc, & \text{åñëè } x \ge b. \end{cases}$$

Значение $x = 0{,}5$ ввести с компонента Edit, результаты вычислений вывести в компонент Memo.

Компонет Memo – это многострочное текстовое окно для ввода или вывода значений данных программы. В окне компонентов C++ Builder, он имеет обозначение, изображенное на рис. 4.1.

Рисунок 4.1 – Обозначение компонета Memo

Основные свойства компонента Memo:

Name – имя компонента в программе;

Lines – окно для ввода или редактирования начальных данных программы,

ScrollBars – установка в окне Memo линейки прокручивания. Это свойство может принимать одно из значений: None, Vertical, Gorizontal, Both (отсутствующая, вертикальная, горизонтальная, обе).

Основные методы компонента Memo:

Clear – очистка окна Memo,

Lines->Add(s) – добавление строки S в окно Memo.

Форма проекта представлена на рис. 4.2.

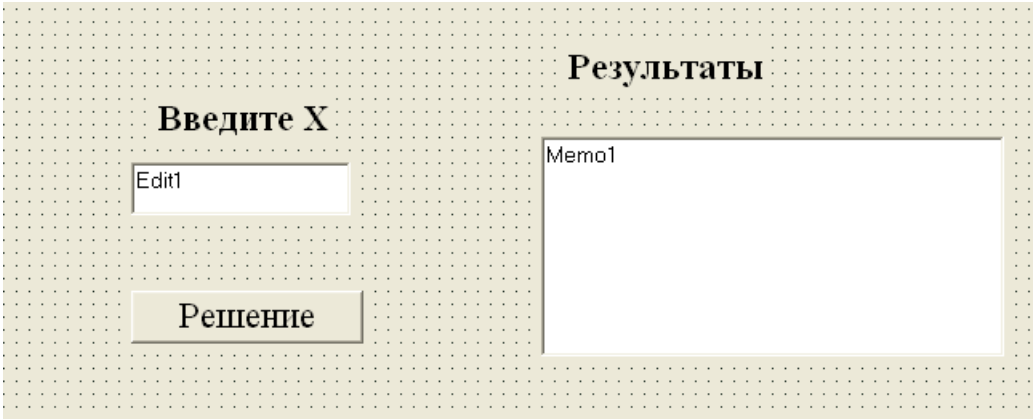


Рисунок 4.2 – Форма проекта

Приводим текст программы для выполнения вычислений:

```
//Обработка события нажатия кнопки “Решение”
void __fastcall TForm1::Button1Click(TObject *Sender)
  {
    float x=StrToFloat(Edit1->Text);
            int s:=1;
   M1:   switch (s)
         {
           case   1: { a=-3.7; b=5.6;  c=fabs(tan(b*x));break;}
           case   2: { a=0.81; b=-2.4; c=fabs(tan(b*x));break;}

           case   3: { a=2.5;  b=0.6;  c=fabs(tan(b*x));break;}
```

```
        }
        if (x<a)   y=sin(exp(a+c))+pow(x,2);
        if ((x>=a) && (x<b))   y=pow((a+fabs(5.3*b)),1/3)+c;
        if (x>=b)  y=pow(cos(a),2)+sin(x*x)-b*c;
      // вывод результатов в Мето
      Memo1->Lines->Add("x="+ FloatToStr(x)+" y=" + FloatToStr(y)
                                  + IntToStr(s)+"й -– вариант");
        if (s<=3)   goto M1;
      }
```

Пример 2. Вычислить значение функции

$$L = \begin{cases} x^2, & n = 1; \\ x / n, & n = 2; \\ x + n, & n = 3; \\ x, & n = 4 \end{cases}$$

в зависимости от значения параметра **n**:

……..

```
{ double L;
    int n;
     n=StrToInt(Edit1->Text);
    switch (n)
        {
        case 1:{L=x*x;break;}
        case 2:{L=x/n;break;}
        case 3:{L=x+n;break;}
        case 4:{L=x;break;}
         }
        ...
// вывод значения L; Edit2→Text=FboatToStr(L);
}
```

Перечень знаний и умений, которые должен приобрести студент в процессе изучения материала данного модуля

Знание:

- Архитектура компьютера;
- Операционные системы;
- Текстовый процессор MS Word;
- Среда программирования Builder;
- Алгоритм, его свойства и средства описания;
- Элементы алгоритмического языка С++ и программы линейной структуры;
- Логические выражения и приоритет выполнения логических выражений;
- Операторы передачи управления в С++.

Умение:

- Выполнять работу с файлами в операционной системе Windows;
- Создавать текстовые документы в редакторе Word с формулами и изображениями;
- Создавать и обрабатывать таблицы в редакторе Word;
- Составлять алгоритмы и программы с линейной и разветвленной структурами в С++ Builder и выполнять их на компьютере.

Задания - тесты для проверки знаний и умений

Дополните утверждение, вписав пропущенное число:
1 байт содержит … бит

Дополните утверждение, вписав пропущенное число:
1 Килобайт содержит … байт

Дополните утверждение, вписав пропущенное числоо:
Для размещения в памяти ПК слова из 7 символов необходимо … бит

Дополните утверждение, вписав пропущенное слово:
Сокращенное название памяти ЭВМ, что доступная только для чтения, а не для записи – это …

Дополните утверждение, вписав пропущенное слово:
Программа, которая противодействует работе вируса и/или возобновляет поврежденные файлы имеет название…

Дополните утверждение, вписав пропущенное слово:
Наименьшее дисковое пространство, которое может быть выделено для отдельного файла, имеет название …

Дополните утверждение, вписав пропущенное слово:
Устройство, которое обеспечивает управление компьютером и выполняет вычисления называется …

Дополните утверждение, вписав пропущенное слово:
Наименьший участок дорожки диска, из которой можно прочитать или записать данные имеет название …

Дополните утверждение, вписав пропущенное слово:
Аппаратный сигнал, который сообщает о наступлении некоторого события во время работы компьютера имеет название …

Дополните утверждение, вписав пропущенное слово:
Аппаратура, которая обеспечивает связь процессора с внешними устройствами имеет название …

Дополните утверждение, вписав пропущенное значение:
Отношение 2*5<=17 % 3
имеет значение …

Запишите на языке С++ оператор вычисления переменной Y

$$Y=\begin{cases}\cos x, & \text{если } 0<x<2\\ 1-\sin x, & \text{в других случаях}\end{cases}$$

Запишите на языке С++ оператор вычисления переменной К,

$$K=\sin x^3+\ln|x|$$

Наберите номер правильного, на Ваш взгляд, ответа:
Стандартное расширение файла текстового процессора Word – это:

1. doc
2. xls
3. txt
4. wrd

Наберите номер правильного, на Ваш взгляд, ответа:
Система программного обеспечения, которая руководит работой всех структурных узлов компьютера, имеет название

1. автоматизированная
2. интеллектуальная
3. операционная

Наберите номера правильных, на Ваш взгляд, ответов:
Буфер обмена в редакторе WORD позволяет выполнять команды

1. Вставить
2. Найти
3. Вырезать
4. Копировать
5. Очистить

Наберите номера правильных, на Ваш взгляд, ответов:
Операторы, которые объявляют переменные целого типа (в С++), – это

1. Var r,j : integer;
2. int r, j;
3. Var c : char;
4. Type t= integer; Var x,y : t;
5. Var m,n, : word;

Дополните утверждение, написав пропущенное значение:
Арифметическое выражение a * b = = (c – d)
где a = 5, в = 4, c = 12, d = 2 (все данные целого типа) имеет значение …

Наберите номер правильного, на Ваш взгляд, ответа:
Оператор, который преобразует переменную действительного типа в строковую, – это

1. k:= Copy(s,n,k);
2. Str (x,k);
3. Delete(s,n,k);
4. FloatToStr(x);

Наберите номер правильного, на Ваш взгляд, ответа:
Совокупность данных, размещенных на диске и имеющих общее имя и назначение – это

1. файл
2. процессор
3. сектор
4. кластер

Запишите на языке C++ оператор присвоения

$$y = \frac{1 + \sin x^2}{\sqrt{5a} + \ln x}$$

Запишите на языке C++ оператор вычисления переменной Y

$$Y = \begin{cases} \sin x, & \text{если } x>0.5 \\ \cos x, & \text{если } x<=0.5 \end{cases}$$

Дополните утверждение, вписав пропущенное слово:
Выражение x1 || x2
где x1 = true, x2 = false имеет значение …

Дополните утверждение, вписав *пропущенное слово:*
Константа 0.2731E3 в фиксированном формате имеет вид …

Дополните утверждение, вписав пропущенное значение:
Логическое выражение 7 + 3 > 16 – 4 * 3 имеет значение…

Дополните утверждение, вписав пропущенное значение:
Логическое выражение (– 3 > = 5) || (7 < 9) && (0 < 3) имеет значение…

Дополните утверждение, вписав пропущенное значение:
После выполнения операторов

```
x =  2.5;
if (x > = 0.5)  z = 7.7;       z = 5.5;
```

переменная z будет иметь значение…

Наберите номера правильных, на Ваш взгляд, ответов:

В приведенном фрагменте программы

```
 nom  = 2 * pow ( 2,2 );
switch (nom)
{
 case  2 : {в  = d; break; }
  case 4 : {в  = d * x; break; }
  case 8 : {в  = d * exp( x ); break; }
 case 16 : {в  =  sin( x )+d; break; }
}
```

Наберите номера правильных, на Ваш взгляд, ответов:

Операторы без ошибок – это

1. if(x <=6) y=2*x ; else y=cos(x);
2. if y<=x then y:= exp(x * в);
3. if a<>0   if b<>0   y:=2*x;
4. if (x>0) y=ln(x) else y=exp(x);