Міністерство транспорту та зв'язку України
Державна адміністрація зв'язку
ОДЕСЬКА НАЦІОНАЛЬНА АКАДЕМІЯ ЗВ'ЯЗКУ
ім. О.С. ПОПОВА

Кафедра інформаційних технологій

Курсова робота
з дисципліни
"Інформатика"

МОДУЛЬ 2

«Організація бібліотек і підпрограм для роботи з масивами»

для студентів заочної форми навчання
усіх спеціальностей

Одеса 2009

УДК 004.43 План НМВ 2009 р.

Укладачі: **Л. М. Буката, С. В. Ніколаєнко**

Схвалено:
на засіданні кафедри
Інформаційних технологій академії
і рекомендовано до друку
протокол № 3 від 24.10.2009 р.

Затверджено
методичною радою академії
протокол № 6 від 10.02.2009 р.

ПРОГРАМА ДИСЦИПЛІНИ “Інформатика”

1 Основні відомості про персональний комп’ютер та його програмне забезпечення

1.1 Архітектура комп’ютера: Універсальна схема ЕОМ. Процесор. Організація оперативної пам’яті. Системи числення. Організація збереження даних. Зображення інформації в ЕОМ

1.2 Операційні системи: Загальні відомості про операційні системи персональних комп’ютерів. Файлова система. Операційна система Windows. Програма “Проводник” для роботи з файлами в системі Windows.

1.3 Текстовий процесор MS Word: Піктографічне меню. Елементи головного меню. Робота з буфером обміну. Форматування тексту та сторінки. Побудова таблиць. Редактор формул Equation.

1.4 Середовище програмування C++Builder*:* Вікна C++Builder*:*та головне меню. Управління компонентами форми. Файли проекту. Структура модуля в C++Builder. Послідовність створення та виконання проекта в C++Builder.

2 Основи програмування

2.1 Алгоритмізація обчислювальних процесів: Алгоритм та його властивості. Графічне зображення алгоритму. Алгоритми основних обчислювальних процесів.

2.2 Програмування лінійних обчислювальних процесів: Структура програмного модуля. Стандартні математичні функції. Типи даних: цілий, дійсний, логічний, символьний, рядковий. Стандартні функції перетворення типів. Математичні вирази. Оператор присвоєння.

2.3 Програмування розгалужених обчислювальних процесів: Логічні вирази. Оператор безумовного переходу. Умовний оператор. Оператор вибору.

2.4 Програмування циклічних обчислювальних процесів: Оператор циклу з параметром. Оператор циклу з передумовою. Оператор циклу з післяумовою. Вкладені цикли. Побудова графіків у C++Builder.

2.5 Масивовий тип: Опис даних масивового типу. Введення та виведення елементів масиву. Програмування сум та добутків елементів масивів. Програмування пошуку найбільшого (найменшого) елементів масиву. Програмування обчислення кількості елементів згідно заданій умові.

2.8 Підпрограми: Модульна структура програм. Формальні та фактичні, глобальні та локальні параметри.

2.9 Бібліотеки в мові С++: Основні системні бібліотеки. Підключення бібліотек до *С++*-програми. Організація персональної бібліотеки.

Основні теоретичні відомості

Багатовимірні масиви

Вимірність масиву

Кількість індексів визначає вимірність масиву, наприклад, вектори в програмах – це одновимірні масиви, матриці – двовимірні. Кількість індексів у елементів масивів є необмежена. Значення індексів записують після імені масиву в квадратних дужках. Наприклад:

b[4][5], Matr [I][J+1], P['F']['K'] – елементи матриць: b_{45}, $Matr_{i,j+1}$, $P_{'F','K'}$;

У пам'яті комп'ютера елементи масиву розміщуються один за одним у такий спосіб, що при переході від молодших адрес до старших першим змінюється крайній правий індекс, а потім всі інші справа наліво. Так само як і одновимірний масив, багатовимірний масив загалом може займати в пам'яті не більше 2 Гбайт.

Розглянемо прямокутну таблицю з *m* x *n* однотипних елементів, що містить рядків та стовпчиків. У математиці таку таблицю називають матрицею, а дані, що їх містить матриця, елементами. У програмуванні матричні структури називають двовимірними масивами.

Описування багатовимірних масивів

Багатовимірні масиви (як і одновимірні) в програмах можна описувати двома способами :

1.

тип ім'я [*розмір 1*] [*розмір 2*] … [*розмір n*];

2.

typedef *тип_даних ім'я_типу* [*розмір 1*] [*розмір 2*] … [*розмір n*];
ім'я_типу ім'я_масиву ;

Кількість елементів масиву дорівнює добутку кількості елементів кожного індексу.

Приклади описування масивів:

int Mas1 [4][4]; // квадратна матриця 4x4 із 16 елементів цілого типу,

float Mas1 [5][5]; // квадратна матриця 5x5 із 25 елементів дійсного типу,

char Mas2 [10][3]; // двовимірний масив із 10x3=30 елементів символьного типу;

double Mas3 [4][5][4]; // тривимірний масив із 4x5x4=100 дійсних елементів.

Для описування багатовимірних масивів зручно використовувати ініціалізацію початкових значень, яка дозволяє водночас описати масив й задати його значення, причому не обов'язково всіх.

Приклади ініціалізації двовимірних масивів:

1) int w[3][3] = { { 2, 3, 4 }, { 3, 4, 8 }, { 1, 0, 9 } };
2) float C[4][3]={1.1, 2, 3, 3.4, 0.5, 6.8, 9.7, 0.9};
3) float C[4][3]={{1.1, 2, 3}, {3.4, 0.5, 6.8},{ 9.7, 0.9}};
4) float C[4][3]={{ 1.1, 2}, {3, 3.4, 0.5},{ 6.8},{ 9.7, 0.9}};

Наприклад, створимо тип з ім'ям matr як масив додатних цілих чисел з 10-ти рядків, 7-ми стовпчиків і опишемо два масиви M1 та M2:

```
typedef unsigned short matr[10];
mass M1,M2;
```

Для описування багатовимірних масивів зручно використовувати також типізовані константи-масиви, які дозволяють водночас описати масив й задати його значення в розділі констант, наприклад:

```
const int  arr[2][5] = {{4, 3},{ -7, 123};
```

Але змінювати значення елементів констант-масивів у програмі недопустимо.

Введення та виведення елементів багатовимірних масивів

Виводити значення масивів можна у файл або на форму, використовуючи різноманітні компонети C++Builder. При цьому, виводити значення елементів масивів можна лише *поелементно,* для чого слід зорганізувати цикли змінювання за значеннями кожного індексу. Як зорганізувати виведення масивів у файл, буде розглянуто далі в інших лабораторних роботах. Тепер розглянемо, як зорганізувати виведення багатовимірних масивів на форму за допомогою компонентів Memo та StringGrid.

За допомогою компонента **Memo**  можна виводити масиви з будь-якою кількістю елементів, оскільки можна використовувати смуги прокручування (надати властивості ScroollBar значення ssBoth або ssVertical).

Приклад виведення значень матриці R[3][4]:

```
Memo1->Clear();
for (i=0;i<3;i++)
 { st="" ;
   for (j=0;j<4;j++)  st+=FormatFloat("0.00",B[i][j])+"   ";
   Memo1->Lines->Add(st);}
```

Виведення масивів за допомогою компонента **StringGrid** (рис.1).

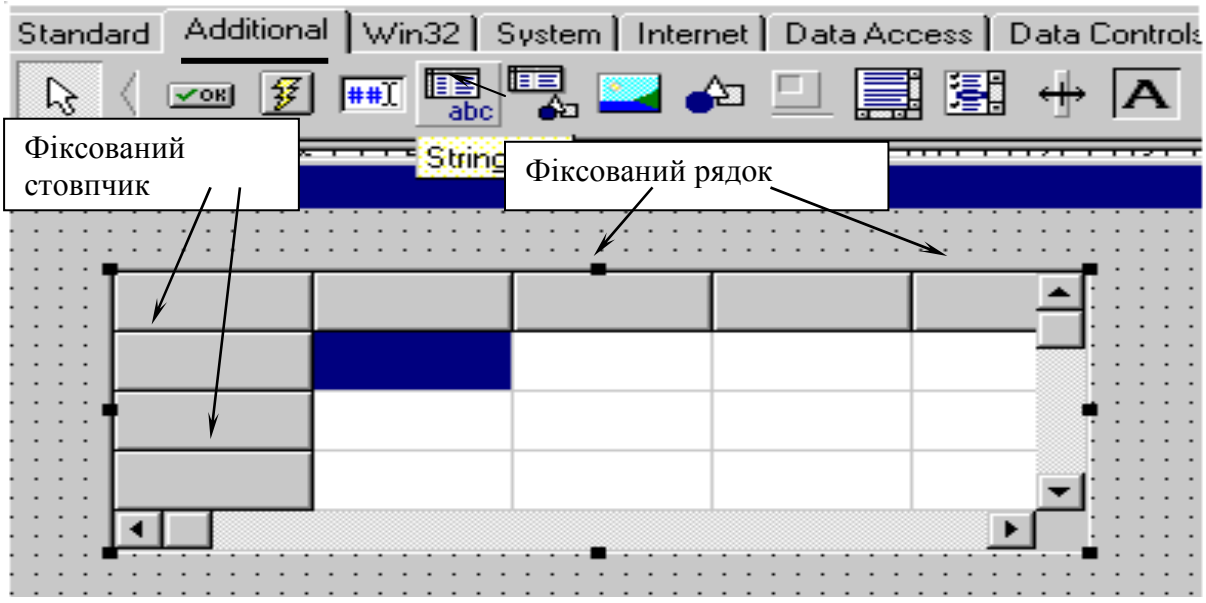

Рисунок 1 – Компонент **StringGrid**

Компонент **StringGrid** має вигляд таблиці з комірками і розташований на вкладці ***Additional*** палітри компонент.

Цей компонент ми будемо використовувати вперше, тому наведемо опис його основних властивостей (табл. 1):

Таблиця 1 – Властивості компонента StringGrid

Властивість	**Призначення**
Name	Ім’я компонента для доступу до його властивостей
ColCount	Кількість стовпчиків таблиці
RowCount	Кількість рядків таблиці
Cells	Масив комірок таблиці. Наприклад, Cells[***i,j***] – це комірка, що розташована на перетинанні ***i***-го стовпчика та ***j***-го рядка (*нумерація починається з нуля*).
FixedCol	Кількість фіксованих стовпчиків (для заголовка)
FixedRow	Кількість фіксованих рядків (для заголовка)
Options**.**goEditing	Ознака дозволу на редагування вмісту комірки
Options**.**Tabs	Ознака дозволу на переміщення за таблицею за допомогою клавіші <Tab>
Options**.**goColSizing	Ознака дозволу на змінювання ширини стовпчиків
Options**.**goRowSizing	Ознака дозволу на змінювання висоти рядків

В кожній комірці (Cells) можна розташувати величину рядкового типу, як і в інших компонентах, якими ми користувалися раніше (Edit, Label та ін.). Властивості групи Options відкривають подвійним клацанням лівої кнопки миші і надають їм значення True або False. Наприклад, для виведення матриці R компоненту StringGrid слід надати такі властивості:

Властивість	**Значення**
Name	mb
ColCount	5
RowCount	4
FixedCol	1
FixedRow	1
Options.goColSizing	True
Options.goRowSizing	True

Для виведення матриці у StringGrid необхідно зорганізувати цикли за номерами рядків та стовпчиків. Приклад фрагмента виведення матриці intB[3][4] у компонент StringGrid1:

```
for (i=0;i<3;i++)
for (j=0;j<4;j++) StringGrid1->Cells[j+1][i+1]=FormatFloat("0.00",B[i][j]);
```

Приклад створення заголовків таблиці у комірках зафіксованих рядка і стовпчика (з індексом 0).

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{ int i,j;
  for (i=1;i<=3;i++)StringGrid1->Cells[0][i]=IntToStr(i)+”-ий рядок”;
  for (j=1;j<=4;j++) StringGrid1->Cells[j][0]=IntToStr(j)+”-ий стовпчик”;
}
```

Вводити значення масивів можна використовуючи такі компоненти, як StringGrid, Memo. Як і при виведенні масивів, при введенні слід зорганізовувати цикли в яких змінюється кожний індекс.

За допомогою компонента **StringGrid** можна вводити масиви лише під час виконання програми.

Приклад введення матриці int B [3][4] з StringGrid1

```
for (i=0;i<4;i++)
for (j=0;j<3;j++)
StringGrid1->Cells[j+][i+1]=FormatFloat("0.00",B[i][j]);
```

можна внести перевірку на незаповнені комірки:

```
for (i=0;i<4;i++)
 for (j=0;j<3;j++)
  if(StringGrid1->Cells[j+1][i+1] != "")
B[i][j]=StrToFloat(StringGrid1->Cells[j+1][i+1]);
  else { ShowMessage("Введіть у ["+IntToStr(i+1)+","+
IntToStr(j+1)+"] елемент"); break;  }
```

Приклад програми з двовимірним масивом

Завдання. Скласти схему алгоритму та проект програми для введення за допомогою компонента StringGrid елементів матриці **F** дійсних чисел розміром 3*5 й визначити:

– максимальний елемент матриці та його місцезнаходження.

Схема алгоритму розрахунків наведена на рис. 2, результати обчислень надані на рис.3.

На формі розташуємо компоненти StringGrid1, Edit1, Button1, Button2, Button3, Label1 та Label2, а їх властивості вказані у табл. 2.

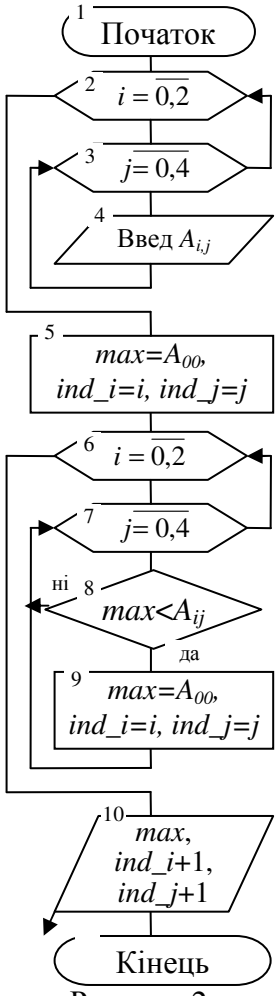

Рисунок 2 – Блок-схема

Таблиця 2 - Властивості компонентів

Компоненти	Властивості	Нові значення
StringGrid1	ColCount	6
StringGrid1	RowCount	4
StringGrid1	Options.goEditing	True
StringGrid1	Options.goTabs	True
Button1	Caption	Решение
Button2	Caption	Очистка
Button3	Caption	Выход
Form1	Caption	Пример 1

Рисунок 3 – Вікно форми з результатами

Текст програми:

```
void __fastcall TForm1::FormCreate(TObject *Sender) // Створення форми
{ for (int i=1;i<=3;i++)StringGrid1->Cells[0][i]=IntToStr(i)+"-й рядок";
  for (int j=1;j<=5;j++)StringGrid1->Cells[j][0]=IntToStr(j)+"-й стовпчик";
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{ float A[3][5],max;   int i, j, ind_i, ind_j;
  for (i=0;i<3;i++)
  for (j=0;j<5;j++)
     if(StringGrid1->Cells[j+1][i+1] != "")
        A[i][j]=StrToFloat(StringGrid1->Cells[j+1][i+1]);
      else
{ ShowMessage("Введіть ["+IntToStr(i+1)+","+ IntToStr(j+1)+"] елемент");
   break;  }
 max=A[0][0]; ind_i=0; ind_j=0;
 for (i=0; i<3; i++)
 for (j=0; j<5; j++)
    if(max < A[i][j])  {max=A[i][j]; ind_i=i; ind_j=j;}
Edit1->Text=FormatFloat("0.00",max)+"   "+IntToStr(ind_i+1)+"- й рядок "+
                                                    IntToStr(ind_j+1)+"-й стовпчик ";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender) // Очистка
{ int i,j;
  for (i=0; i<3; i++)
  for (j=0; j<5; j++)  StringGrid1->Cells[j+1][i+1]="";
  Edit1->Clear();     }
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender) // Вихід
{  Close();  }
```

Функції -підпрограми

В процесі розв’язання складних задач часто виникає необхідність у багаторазовому обчисленні за одними й тими ж виразами за різних початкових даних. В таких випадках для більш ефективного програмування обчислень, які багаторазово повторюються, оформлюються у вигляді самостійного програмного модуля, який багаторазово, в міру необхідності, може бути використаний. Така автономна частина програми, яка реалізує певний алгоритм і дозволяє звертатись до неї з різних частин основної (головної) програми, називається *функцією (підпрограмою)*. Підпрограми підвищують продуктивність праці, економлять пам’ять, роблять всю програму більш наочною і спрощують розроблення великих програм. Кожна підпрограма визначається лише один раз, а виконуватись може багаторазово.

Використання функцій веде до спрощення структури програми, дозволяє уникнути написання аналогічних частин коду, оскільки функцію записують один раз, а викликати її на виконання можна багато разів з різних точок програми і з різними аргументами.

У загальному випадку функція задається своїм ім'ям. Вона може приймати параметри і повертати значення.

Функція починає виконуватися у момент *виклику*. Будь-яка функція повинна бути *оголошена* і *визначена*. Оголошення функції повинне знаходитися в тексті раніше її виклику для того, щоб компілятор міг здійснити перевірку правильності виклику.

В *об'явленні* функції (*прототипи*, *заголовку*) описується її інтерфейс. Він містить всі дані про те, яку інформацію повинна одержувати функція (список формальних параметрів) і яку інформацію вона повертає. *Визначення функції* містить, окрім оголошення, *тіло* функції, що є послідовністю операторів і описів у фігурних дужках:

тип ім'я ([список_параметрів])
{тіло_функції}

1. Тип значення, що повертається, може бути будь-яким, окрім масиву і функції (але може бути покажчиком на масив або функцію). Якщо функція не повинна повертати значення, указується тип void.
2. Список параметрів визначає величини, які вимагається передати у функцію при її виклику. Елементи списку параметрів розділяються комами. Для кожного параметра, переданого у функцію, указуються тип і ім'я (у оголошенні імена можна опускати).
3. Тіло функції – дії, які виконує функція.

Повернення значення з функції у функцію, яка викликає її, реалізується оператором

return вираз;

Якщо функція описана як void, вираз не указується.

Приклади:

int f1( ) {return 1;} *//правильно – функція типу int повертає* 1
void f2( ) {return 1;} *//неправильно,* f2 *не повинна повертати значення*
double f3 {return 1;} *//правильно,* 1 *перетвориться до типу double*

Щоб використовувати функцію, не треба знати, як вона працює, – достатньо знати, як її викликати. Для виклику функції потрібно вказати її ім'я, за ним в круглих дужках через кому перераховуються аргументи, які передаються. У ролі фактичних параметрів можуть виступати відповідно до їх типу константи, змінні або вирази. Виклик функції може знаходитися в будь-якому місці програми, де за синтаксисом допустимий вираз того типу, який формує функція. Якщо тип значення, що повертається, не void, вона може входити до складу виразів або, в окремому випадку, розташовуватися в правій частині оператора присвоєння.

Параметри, перелічені в заголовку опису функції, називаються *формальними*, а записані в операторі виклику функції – *фактичними*. При виклику функції в першу чергу обчислюються вирази, що стоять на місці фактичних параметрів; потім в стеку виділяється пам'ять під формальні параметри функції відповідно до їх типу і кожному з них привласнюється значення відповідного фактичного параметра. При цьому перевіряється

відповідність типів і, при необхідності, виконуються їх перетворення. При невідповідності типів видається діагностичне повідомлення.

У визначенні, в оголошенні і при виклику однієї і тієї ж функції потрібно дотримуватися наступних правил щодо параметрів:

- порядок проходження параметрів повинен бути однаковим;
- тип відповідних параметрів у визначенні і оголошенні повинен бути однаковим;
- типи фактичних параметрів повинні співпадати з типами відповідних формальних параметрів або бути такими, щоб можна було їх неявно перетворити до типу формальних параметрів.

Приклади стандартних функцій

1. Функція піднесення до степеня pow(x,n). Вона оголошена таким чином:

   double pow(double x, double y);

2. Ця функція має два параметри – *x* та *y* і повертає значення. При виклику функції замість змінних *x* і *y* можуть бути підставлені дійсні або цілі числа, в іншому випадку вони неявно будуть перетворені до типу *double*.
3. Функція обчислення модуля цілого числа abs(int x). Вона оголошена таким чином:

   int abs(int x);

4. Функція має один цілий параметр і повертає ціле значення. Як параметр може бути підставлене ціле або дійсне число. У іншому випадку воно буде перетворене в ціле (відкинута дробова частина).

Приклад функції користувача, повертаючої суму двох цілих величин:

```
int sum (int a, int b);//оголошення функції (прототип:ім'я функції sum,
      //параметри – два цілі числа, значення, яке повертається, ціле)
int sum(int a, int b){      //визначення функції
return (a+b);}
```

Наприклад, оголошені змінні:

```
int a=2, b=3, c,d,x;
```

Можливі наступні виклики цієї функції:

```
x=sum(6,a);          //виклик функції як перший параметр
                  //підставляється число 6 як другий – значення змінної a,
                     //тобто 2. В результаті змінної x привласнюється 8

c=sum(a,b);          // виклик функції як перший параметр
//підставляється значення змінної a, тобто 2 значення
//змінної b, тобто 3. У результаті змінної x привласнюється 5

d=sum(c-2, a*b);  // виклик функції, яка обчислює суму c-2 і a*b,
                     //тобто чисел 3 і 6 (d=9)
```

Всі величини, описані всередині функції, а також її параметри, є *локальними*. Це означає, що іншим функціям даної програми про них нічого не

відомо. Областю їх дії є функція, тобто поза функцією вони не можуть бути використані. Пам'ять під локальні змінні, які використовуються у функції, виділяється тільки на час виконання функції. Після завершення роботи функції пам'ять очищається, значення змінних не зберігаються.

Глобальні змінні оголошуються в головній програмі, тобто після підключення заголовочних файлів до всіх функцій. Їх видно у всіх функціях, де не описані локальні змінні з тими ж іменами, тому використовувати їх для передачі даних між функціями дуже легко. Проте це не рекомендується, оскільки утруднює редагування програми. Потрібно прагнути, щоб функції були максимально незалежні, а їх інтерфейс повністю визначався прототипом функції.

Існує два способи передачі параметрів у функцію: за значенням і за адресою.

При передачі за значенням в стек заносяться копії фактичних параметрів, і оператори функції працюють з цими копіями. Доступу до початкових значень параметрів у функції немає, а, отже, немає і можливості їх змінити.

При передачі за адресою в стек заносяться копії адрес параметрів, а функція здійснює доступ до елементів пам'яті за цими адресами і може змінити початкові значення параметрів.

Розглянемо наступну функцію:

```
void f(int i, int* j)
 { i++;  (*j)++;}
```

Перший параметр (*i*) передається за значенням. Його зміна у функції не впливає на початкове значення змінної *i*. Другий параметр (*j*) передається за адресою за допомогою покажчика. Функція має доступ для зміни параметра *j* і не має доступу для зміни параметра *i*. Приклади виклику цієї функції:

```
int x=3, y=5, *z(7);
f(3, &y);     // У функцію передається число 3 як перший параметр і
              // адреса змінної y. Після виходу з функції y прийме значення 6.
f(x, &y);     // У функцію передається копія значення змінної x, тобто
              // число 3 і адреса змінної y. Після виходу з функції значення
              // змінної x не зміниться (залишиться рівним 3), а y прийме
              // значення 7.
f(1, z);            // У функцію передається число 1 як перший параметр і
              // змінна z, в якій зберігається адреса деякої цілої змінної
              // із значенням 7. Після виходу з функції значення змінної, на
              // яку указує z, стане рівним 8.
```

Якщо вимагається заборонити зміну параметра усередині функції, використовується модифікатор *const*:

```
int f(const int*);
```

За умовчанням параметри будь-якого типу, окрім масиву і функції, вдаються у функцію за значенням.

При використовуванні масиву, як параметра, у функцію передається покажчик на його перший елемент, іншими словами, масив завжди передається за адресою. При цьому, якщо масив має змінну розмірність, інформація про

кількість елементів втрачається, і слід передавати його розмірність через окремий параметр, якщо вона не є постійною.

```
int sum(const int* mas, const int n);          // прототип функції
int sum (const int* mas, const int n){         //визначення функції
    //варіанти:          int sum(int mas[], int n)
    //або      int sum(int mas[n], int n)
    //(величина n повинна бути константою)
  int s = 0;
  for (int i = 0; i < n; i++) s += mas[i];
 return s; }
```

Виклик функції:

```
int const n = 10;
int marks[n]= {3, 4, 5, 4, 4};
s=sum(marks, n);// виклик функції
return 0;
}
```

Аналогічна програма для масиву з 5-ти елементів (кількість відома наперед):

```
int sum(const int mas[5]);              //прототип функції
int sum (const int mas[5]){             //визначення функції
int s = 0;   for (int i = 0; i < 5; i++) s += mas[i];
return s;}
```

Виклик функції:

```
int marks[5]= {3, 4, 5, 4, 4};
s=sum(marks);
return 0;
```

Наведемо приклади функцій:

Обчислимо значення математичної функції *sign(x)*, яка задається таким чином:

$$sign(x) = \begin{cases} -1, \text{якщо } x < 0 \\ 0, \text{якщо } x = 0 \\ 1, \text{якщо } x > 0 \end{cases}$$

Функція в програмі мовою С++ матиме наступний вигляд:

```
int fsign(float x);          //прототип функції
int fsign(float x){          //визначення функції:
  if (x<0) return (-1);
    else if (x==0) return (0);
      else return (-1);
}
```

Виклик функції:

```
int y=fsign(5);
```

В даному випадку замість *x* підставляється число *5*, функція повертає значення 1 (оскільки 5>0).

```
int a=-2;
int z= fsign(a);
```

В цьому випадку замість *x* підставляється значення змінної *a*, виконується неявне перетворення цілого числа в дійсне, функція повертає значення -1 (оскільки -2<0).

Обчислимо значення факторіалу деякого цілого числа. Факторіал цілого числа n – це добуток всіх чисел від 1 до n.

Функція в програмі мовою С++ матиме наступний вигляд:

```
float fact(int n);              //прототип функції
                                //визначення функції:
float fact(int n){
        int i;
        float p=1;
        for (i=1; i<n; i++)
                p=p*i;
        return p;
}
```

Зверніть увагу, що функція fact повертає значення типу float, оскільки для аргументів, які більше *7*, ціле значення факторіалу перевищує 32767 і виходить за межі цілого типу.

Виклик функції:

```
int y=fact(5);
```

В даному випадку замість *x* підставляється число 5, функція повертає значення 120, (оскільки 1*2*3*4*5=120).

Організація бібліотек функцій

Бібліотекою називається файл, який містить декілька функцій і зорганізований таким чином, що кожну з цих підпрограм можна вилучити з бібліотеки і приєднати до будь-якої „зовнішньої” програми. Окрім функцій, бібліотеки можуть містити оголошення констант, змінних і типів, якими теж може користуватися будь-яка зовнішня програма.

Розрізняють так звані *системні* бібліотеки, які входять до складу системи Builder, і *власні* бібліотеки (або *бібліотеки користувача*), які можна створювати самому.

Бібліотека користувача складається з двох файлів. Один з них називається *заголовочним*. Він може містити визначення типів, констант, вбудованих функцій, оголошення функцій, даних, директиви препроцесора, коментарі тобто *інтерфейс*. Заголовочні файли мають розширення *.h*. Другий файл містить *визначення* функцій, прототипи яких вказані в заголовочному файлі тобто реалізацію. Він має розширення .cpp.

Щоб в програмі можна було користуватися типами і функціями, оголошенними в певній бібліотеці, потрібно скористатися директивою *#include*, вказавши ім'я відповідного заголовочного файла.

#include "ім'я заголовочного файла"

Для включення файлів із стандартних каталогів, потрібно замість лапок використовувати кутові дужки < *i* >.Наприклад:

#include <stream.h> //підключення із стандартного каталога

#include "myheader.h" //підключення із поточного каталога

Послідовність дій при створенні файлів бібліотеки:

1. Виконати команду File/New/Unit. Створиться порожній модуль без пов'язаного з ним вікна форми (рис. 1).

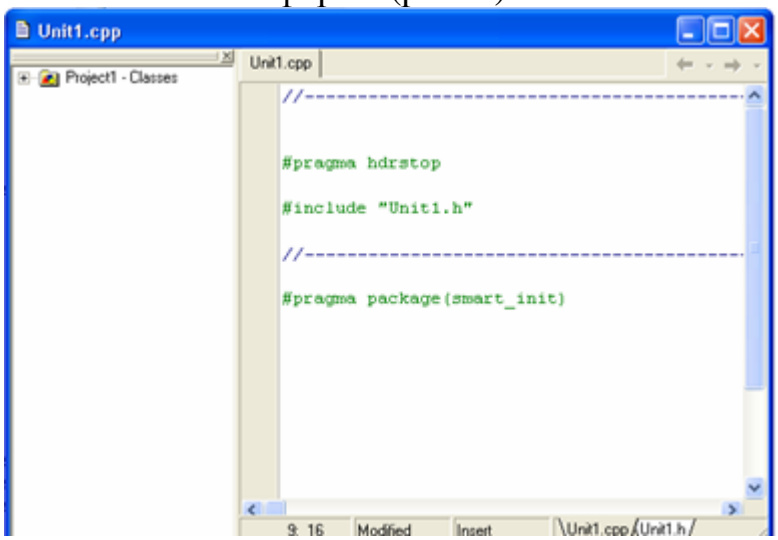

Рисунок 1- Файл реалізації

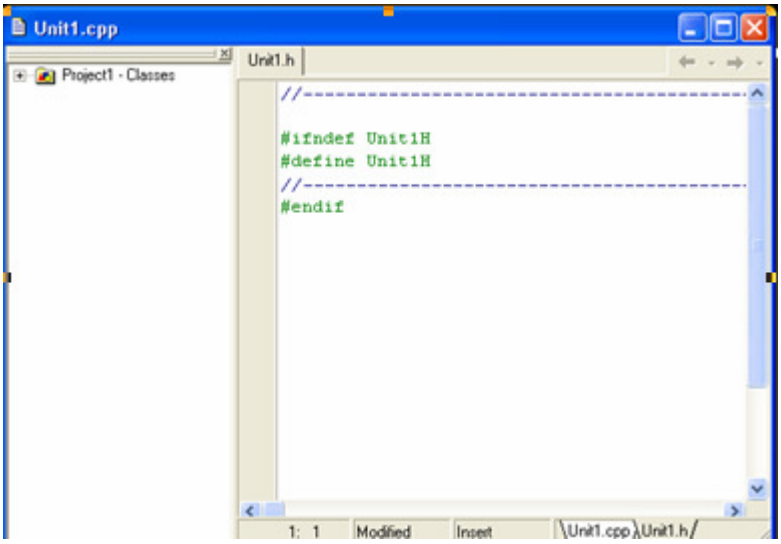

Рисунок 2 - Заголовочний файл

На рис. 1. показаний вміст файлу *Unit1.cpp*, а на рис. 2. – вміст відповідного заголовочного файлу.

2. Зберегти файл на диску з ім'ям створеної бібліотеки. Unit1 в обох файлах автоматично заміниться на нове ім'я. Наприклад, задамо ім'я Mybibl.

3. У файл Mybibl.h ввести необхідні оголошення і заголовки функцій.
4. У файл Mybibl.cpp ввести визначення функцій, заголовки яких знаходяться у файлі Mybibl.h.
5. Для підключення створеної бібліотеки до програми виконати команду *File/Include Unit Hdr…* і вказати ім'я *заголовочного* файла.

Приклад виконання завдання

1. У бібліотеці за допомогою підпрограм виконати розрахунок елементів матриці *a*, що складається з 6-ти рядків і 6-ти стовпців за формулою
$$a_{i,j} = \sin i + \cos j, \text{ де } i = 0,2, \ldots,5; j = 0,2, \ldots,5.$$
2. Сформувати одновимірний масив x, як суму елементів рядків матриці $a_{i,j,}$.
3. Функція G – елемент матриці $a_{i,j,}$, найнаближеніший до значення середньоарифметичного елементів матриці $a_{i,j}$.

Порядок виконання завдання

1. Запустимо C++ Builder.
2. Помістимо на форму необхідні компоненти (рис. 3) і задамо їм наступні властивості, які наведено у табл. 1.

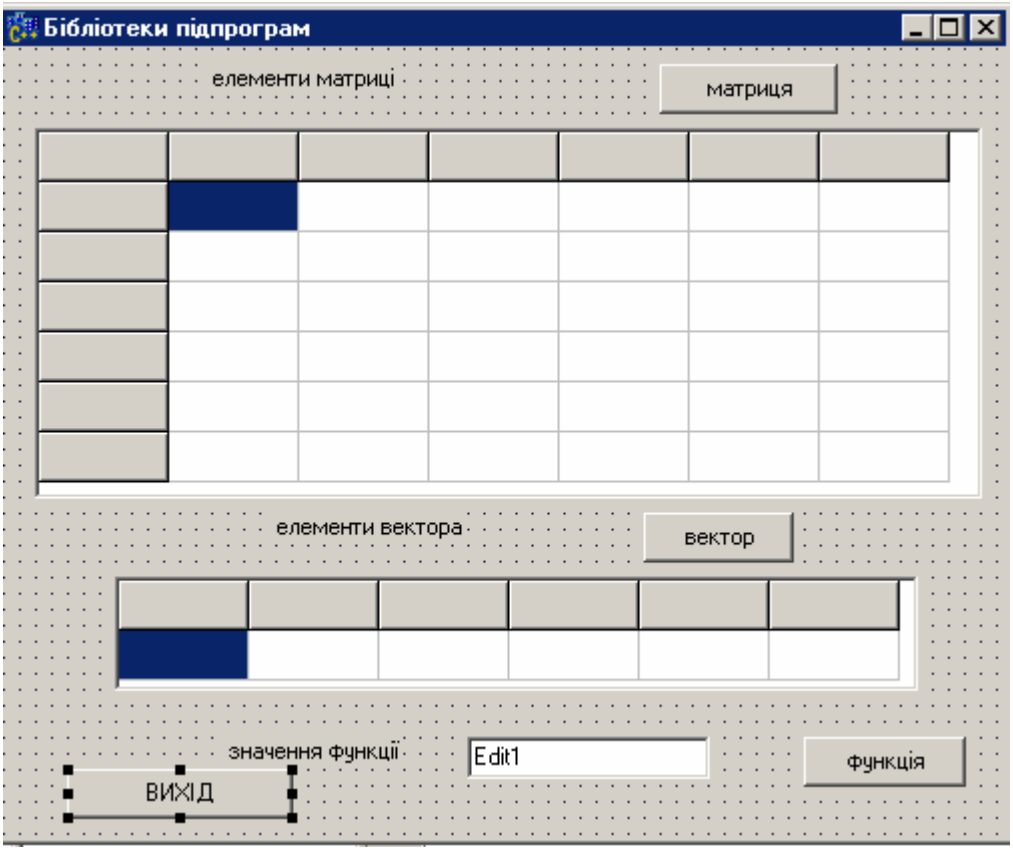

Рисунок 3 – Вікно форми

Таблиця 1 - Властивості компонент

Компонент	Властивість	Значення
Form1	Caption	Бібліотеки підпрограм
Label1	Caption	Елементи матриці
Label2	Caption	Елементи вектора
Label3	Caption	Значення функції
StringGrid1	Name	SG1
StringGrid1	FixedCols	1
StringGrid1	FixedRows	1
StringGrid1	ColCount	7
StringGrid1	RowCount	7
StringGrid2	Name	SG2
StringGrid2	FixedCols	0
StringGrid2	FixedRows	1
StringGrid2	ColCount	6
StringGrid2	RowCount	2
Button1	Caption	Матриця
Button2	Caption	Вектор
Button3	Caption	Функція
Button4	Caption	Вихід

Виконаємо команду File/New/Unit. Створиться порожній модуль без пов'язаного з ним вікна форми. Збережемо проект у теці. При збереженні файлу Unit2 замінимо його ім'я на Mybibl.

У файл Mybibl.h введемо необхідні оголошення і заголовки функцій.

```
//---------------------------------------------------------------------------
#ifndef MybiblH
#define MybiblH
//---------------------------------------------------------------------------
#endif
//глобально оголошуємо змінні
    const       int k=6;
     // оголошуємо типи
   typedef double matr[k][k];
   typedef double vekt[k];
//прототипи функцій
    void elem_Matr (matr c);
    void elem_Vect (matr c,vekt v);
    double G(matr c);
```

У файлі Mybibl.cpp прописуємо безпосередньо визначення функцій, прототипи яких знаходяться у файлі Mybibl.h.

```
//---------------------------------------------------------------------------
#pragma hdrstop
#include "math.h"
#include "Mybibl.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)

//визначення функції розрахунку елементів матриці a

 void elem_Matr (matr c)
{  int i,j;
for (i=0;i<k;i++)
for (j=0;j<k;j++)
c[i][j]= sin(i)+cos(j) ;
  }

//визначення функції розрахунку елементів вектора x
  void elem_Vect (matr c,vekt v)
 {   int i,j;
for (i=0;i<k;i++)
  {
     v[i]=0;
     for (j=0;j<k;j++)
     v[i]+=c[i][j];
   }
}


//визначення функції розрахунку значення G
       double G(matr c)
 {
   int ni=0,nj=0,i,j;
   double   sr=0;
      for (i=0;i<k;i++)
   for (j=0;j<k;j++)
   sr+=c[i][j]/pow(к,2);
         for (i=0;i<k;i++)
     for (j=0;j<k;j++)
   if(fabs(sr-c[i][j])<fabs(sr-c[ni][nj])){ni=i;
     nj=j;
  }
   return c[ni][nj];
  }
```

Переходимо до опису безпосередньо головного модуля (Unit1.cpp).

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
// підключаємо бібліотеку
     #include <Mybibl.h>
#include "Unit1.h"

//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------------------------------

__fastcall TForm1::TForm1(TComponent* Owner)
       : TForm(Owner)
{
}

//---------------------------------------------------------------------------


//глобально оголошуємо змінні
      matr a;   vekt x;

void __fastcall TForm1::FormCreate(TObject *Sender)
{
// Заповнюємо фіксовані комірки компонентів StringGrid
 for (int i=1;i<=k;i++)
 {   SG1->Cells[0][i]="строка №"+IntToStr(i);
    SG1->Cells[i][0]="столб. №  "+IntToStr(i);
    SG2->Cells[0][i-1]="Эл-нт №"+IntToStr(i); }
}
    //---------------------------------------------------------------------------

// «Вихід»
void __fastcall TForm1::Button4Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------------
```

```
// «Виведення елементів матриці»

void __fastcall TForm1::Button1Click(TObject *Sender)
{ // виклик підпрограми обчислення елементів матриці
 elem_Matr ( a) ;
int i,j;
for (i=0;i<k;i++)
for (j=0;j<k;j++)
SG1->Cells[j+1][i+1]=FormatFloat("0.000",a[i][j]);
  Button2->Visible =1;
Button3->Visible =1;
}
//---------------------------------------------------------------------------

// «Виведення елементів вектора»
void __fastcall TForm1::Button2Click(TObject *Sender)
{
// виклик підпрограми обчислення елементів вектора
elem_Vect(a,x);
 for (int i=0;i<k;i++)
SG2->Cells[i][1]=FormatFloat("0.000",x[i]);
}
//---------------------------------------------------------------------------

// «Виведення значення функції G»
void __fastcall TForm1::Button3Click(TObject *Sender)
{
Edit1->Text=FormatFloat("0.000",G(a));
}
//---------------------------------------------------------------------------
```

Завдання до курсової роботи

з теми “Масиви і підпрограми”

1. Обчислити елементи квадратної матриці a_{ij} (i=0,2,…4; j=0,2,…4) за заданою у табл. 6 формулою (стовпчик 2).

2. Використовуючи елементи матриці a_{ij} обчислити елементи вектора $x=\{x_i\}$(i=0,2,…4), згідно з заданим у табл.6 алгоритмом (стовпчик 3).

3. Обчислити значення функції G згідно формулі у табл.6 (стовпчик 4).
   Примітка. Номер варіанта індивідуального завдання відповідає номеру студента в списку групи або визначається викладачем при видачі завдання на курсову роботу.

Вказівки щодо оформлення курсової роботи

1. Курсова робота оформлюється тільки на окремих пронумерованих аркушах, написаних від руки або надрукованих, які підшиті в швидкозшивач.
2. Слід заповнювати тільки одну сторону аркуша з полями для зауважень викладача.
3. Для курсової роботи треба записати (дивись приклад):
   а) умову задачі з індивідуальним завданням;
   б) схему алгоритму розв’язання задачі;
   в) форму проекту та значення властивостей її компонентів;
   г) тексти підпрограм;
   д) результати розрахунків (після виконання програми в дисплейному класі).
4. Схеми алгоритмів повинні бути виконані олівцем під лінійку у відповідності з ЄСПД. Опис алгоритму повинен відображати змістове призначення, порядок слідування та дії операторів.
5. Наприкінці роботи треба вказати список використаної літератури, поставити підпис та дату виконання роботи.

Вказівки до виконання роботи

1. Для обчислення елементів матриці і вектора використовувати підпрограми процедури.
2. Для обчислення значення функції G використовувати підпрограму-функцію.
3. Завдання виконати в середовищі C++ Builder мовою програмування C++.
4. Виведення значень матриці та вектора виконати в компонент StringGrid, значення функції – в компонент Edit або Label.

Таблиця 6 – Варіанти індивідуальних завдань до курсової роботи

№ варіанта	Формула для обчислення елементів матриці	Алгоритм здобуття елементів вектора	Формула для обчислення функції $G(x_0,\dots x_4)$
1	2	3	4
1	$a_{ij}=4^{-(i+j)}\times 3^{i+j}\times\cos(\frac{i\times j}{6})$	Сума додатних елементів рядків матриці	$G=\cos\left(\sum_{i=0}^{4}\left(x_i+\prod_{k=0}^{i}x_k\right)\right)$
2	$a_{ij}=3^{-j+1}\times\left(i-\frac{(j+1)}{3}\right)\times\frac{i+3}{j+6}$	Найбільші елементи стовпців матриці	$G=\ln\left\|\prod_{i=0}^{4}\left(x_i+\sum_{k=0}^{i}x_k\right)\right\|$
3	$a_{ij}=e^{-2(j+1)}+\frac{3(i+1)}{j+1}$	Квадрати найбільших елементів рядків матриці	$G=\sum_{i=0}^{4}\frac{\prod_{k=0}^{i}(x_k+\sin(x_k))}{x_i^2}$
4	$a_{ij}=2^{j}\times(i-4.4)\times\lg\left\|\frac{i+j+1}{3}\right\|$	Сума від'ємних елементів рядків матриці	$G=\sum_{i=0}^{4}\frac{\sin\left(\prod_{k=0}^{i}x_k\right)}{x_i}$
5	$a_{ij}=\frac{3+i}{4+j}\times 2(i+1)^{-2}+\frac{4(i+1)}{j+1}$	Найменші елементи стовпців матриці	$G=\prod_{i=0}^{4}\left(x_i+\sum_{k=0}^{i}\ln\left\|x_k\right\|\right)$
6	$a_{ij}=2^{j-1}\times\left(\left\|i-3\right\|-1.5\right)+\lg(i+j+1)$	Добуток елементів парних стовпців матриці	$G=\sum_{i=0}^{4}\frac{\prod_{k=0}^{i}(x_k-1.75)}{x_i}$
7	$a_{ij}=\sqrt{i+5}\times 2(j+1)^{-2}-\left\|i+j-4\right\|$	Добуток додатних елементів рядків матриці	$G=\prod_{i=0}^{4}\frac{x_i}{\sum_{k=0}^{i}(x_k+\ln(x_k))}$
8	$a_{ij}=2^{i+j}\times\left(i+j-(j+1)\times\cos\left(\frac{i+1}{j+1}\right)\right)$	Елементи головної діагоналі матриці	$G=\lg\left\|\sum_{i=0}^{4}\left(x_i^2+\prod_{k=0}^{i}x_k^{-2}\right)\right\|$
9	$a_{ij}=(i+1)^{-3}\times\sqrt{2(j+1)-1.5}-\frac{2+j}{\left\|2-j\right\|+1.2}$	Сума елементів головної діагоналі і першого рядка матриці	$G=\prod_{i=0}^{4}\left(x_i-\sqrt{\sum_{k=0}^{i}\left\|x_k\right\|}\right)$
10	$a_{ij}=2^{-i+1}\times(j-3.8)\times\left((j+1)\times\left\|2-\frac{3}{i+1}\right\|\right)$	Добуток від'ємних елементів рядків матриці	$G=\sum_{i=0}^{4}\left(\sqrt[3]{\prod_{k=0}^{i}x_k}+x_i^2\right)$
11	$a_{ij}=3.5^{-2(j+1)}\times\left\|i-3(j+1)\right\|+\frac{2}{i+j+1}$	Елементи неголовної діагоналі матриці	$G=\prod_{i=0}^{4}\left(\cos\left(\sum_{k=0}^{i}x_k\right)+\frac{x_i}{2i}\right)$

№ варіанта	Формула для обчислення елементів матриці	Алгоритм здобуття елементів вектора	Формула для обчислення функції $G(x_0,\dots x_4)$
1	2	3	4
12	$a_{ij}=4^{-2+i}\times\left(i-\frac{i+1}{4+j}\right)\times\sin\left(4(i+1)\right)$	Сума елементів неголовної діагоналі і другого стовпця матриці	$G=\sum_{i=0}^{4}\left(\sqrt{x_i}+\prod_{k=0}^{i}\frac{x_k}{k^2}\right)$
13	$a_{ij}=\frac{i+(j+1)^{-2}}{2+i+j}\times 2(i+1)e^{-j}$	Сума елементів першого й п'ятого рядків матриці	$G=\sum_{i=0}^{4}\frac{2.75\,i+x_i}{\prod_{k=0}^{i}\left(k^2+x_k\right)}$
14	$a_{ij}=2{,}5(i+1)-\frac{4(i+1)+j}{\sqrt[3]{\lvert i-j+1\rvert}+(i+1)^2}$	Добуток додатних елементів стовпців матриці	$G=\sum_{i=0}^{4}(\sqrt[3]{x_i}+\prod_{k=0}^{i}\frac{x_k}{k+e^k})$
15	$a_{ij}=2^{-j+i}\left(\lvert i-4(j+1)\rvert-2\right)+\left(4+i\right)$	Сума елементів головної і неголовної діагоналі матриці	$G=\sqrt[3]{\sum_{i=0}^{4}\left(x_i^2+\prod_{k=0}^{i}x_k\right)}$
16	$a_{ij}=\frac{1{,}5}{i+1}\times\lvert 3-j-i\rvert-5\times\sqrt{\lvert 4-i-j\rvert}$	Добуток елементів другого і четвертого стовпців матриці	$G=\sin\left(\sum_{i=0}^{4}x_i-\prod_{k=0}^{i}x_k\right)$
17	$a_{ij}=\sqrt[3]{i+j+1}\times\sin\left(\frac{j+1}{2}\right)$	Середнє арифметичне елементів стовпців матриці	$G=\lg\left\lvert\prod_{i=0}^{4}\left(x_i-\sum_{k=0}^{i}x_k\right)\right\rvert$
18	$a_{ij}=\sqrt{(i+1)^2+j}\times(i+1)^{-2}\sin\left(i+2\right)$	Різниця елементів головної діагоналі і другого рядка матриці	$G=\cos\left(\sum_{i=0}^{4}\frac{\prod_{k=0}^{i}(x_k-25)}{x_i-0.5}\right)$
19	$a_{ij}=\sqrt{(i+1)^2+(j+1)^2}\times\ln\left(3+j\right)$	Добуток елементів третього і п'ятого рядків матриці	$G=\sum_{i=0}^{4}\frac{\cos\left(\prod_{k=0}^{i}(x_k-0.5)\right)}{x_i-1.2}$
20	$a_{ij}=\ln\left(2+\lvert i-j\rvert\right)+e^{-\sqrt{i+j+1}}$	Різниця елементів другого й четвертого рядків матриці	$G=\prod_{i=0}^{4}\left(x_i-\sum_{k=0}^{i}\lg\lvert x_k\rvert\right)$
21	$a_{ij}=2(i+1)^2\left(\sqrt{i+1}+\sqrt{j+1}\right)\times e^j$	Різниця елементів головної діагоналі й третього стовпця матриці	$G=\sum_{i=0}^{4}\frac{\prod_{k=0}^{i}(x_k+0.15)}{x_i+0.21}$

№ варіанта	Формула для обчислення елементів матриці	Алгоритм здобуття елементів вектора	Формула для обчислення функції $G(x_0,\dots x_4)$
1	2	3	4
22	$a_{ij}=3^{i-j}\times\sqrt{i+j+1}\times\sin(i+1)$	Добуток елементів першого і четвертого стовпців матриці	$G=\prod_{i=0}^{4}\frac{x_i-\lg\lvert x_i\rvert}{\sum_{k=0}^{i}x_k}$
23	$a_{ij}=2^{1-j}\times\lvert i-4\rvert+2\ln(0{,}5+(j+1)^2)$	Сума елементів непарних рядків матриці	$G=\ln\left\lvert\sum_{i=0}^{4}\left(x_i^3-\prod_{k=0}^{i}x_k^{-2}\right)\right\rvert$
24	$a_{ij}=\frac{3(i+1)}{i+j+1}\times\sqrt{\frac{(i+1)^2}{2}}\times(1+j)^2$	Добуток елементів непарних рядків матриці	$G=\sum_{i=0}^{4}\left(\sqrt[3]{x_i}+\prod_{k=0}^{i}\frac{x_k}{k+e^k}\right)$
25	$a_{ij}=\sin\frac{1}{i+j+3}+2\cos^2\left(\frac{i+j+1}{4}\right)$	Найбільші елементи рядків матриці	$G=\sum_{i=0}^{4}\left(\sqrt{\prod_{k=0}^{i}x_k}+\ln\lvert x_i^2\rvert\right)$
26	$a_{ij}=3^{-j}\times(j+\ln\frac{2+(i+1)^2}{i+j+1})$	Різниця елементів першого й п'ятого стовпців матриці	$G=\prod_{i=0}^{4}\left(\sin\left(\sum_{k=0}^{i}x_k\right)+\frac{x_i}{i^2}\right)$
27	$a_{ij}=2.5\times\sqrt[3]{\lvert i+j+1\rvert}+\frac{4(i+j+1)}{2+(j+1)^2}$	Сума елементів неголовної діагоналі і першого стовпця матриці	$G=\sum_{i=0}^{4}\left(\sqrt[3]{x_i}+\prod_{k=0}^{i}\frac{x_k}{k+7.5}\right)$
28	$a_{ij}=2^{-i}\times(j-4.3)+\frac{5(i+1)^2+\sin(j+1)}{e^i}$	Найменші елементи рядків матриці	$G=\sum_{i=0}^{4}\frac{\prod_{k=0}^{i}(k^2-x_k)}{2.3i+x_i}$
29	$a_{ij}=\frac{5-j}{i+1}\times3.1\ln\sqrt{(i+2)^2}\times(j+1)^{-3}$	Сума елементів парних рядків матриці	$G=\prod_{i=0}^{4}\left(x_i-\frac{k^3}{\sum_{k=0}^{i}(k\times x_k)}\right)$
30	$a_{ij}=\ln((i+1)^2+(j+1)^3)-\lg(e^{-j+1})$	Різниця найбільшого і найменшого елементів рядків матриці	$G=\sqrt{\sum_{i=0}^{4}\left\lvert x_i^2-\prod_{k=0}^{i}x_k\right\rvert}$

ПРИКЛАД ОФОРМЛЕННЯ ПРОЕКТУ в C++Builder

Завдання

1. Обчислити елементи матриці за формулою:

$a_{i,j} = \sin(i+1) + \ln(j+1)^3$, де $i = 0,2, \ldots,4$; $j = 0,2, \ldots,4$.

2. Сформувати вектор $\boldsymbol{x}$ $(x_0, x_1, \ldots, x_4)$, як суми квадратів відповідних елементів рядків.

3. Обчислити скаляр за формулою

$$G = \sqrt[4]{\left| \sum_{i=0}^{4} \left(x_i^4 - \prod_{k=0}^{i} x_k \right) \right|}.$$

Примітка. Обчислення до задач 1 і 2 завдання зорганізувати за допомогою підпрограм-процедур, до задачі 3 – за допомогою підпрограми-функції.

Розв'язання задачі

Схеми алгоритмів

Схеми алгоритмів підпрограм розрахунків наведено на рис. 1– 3.

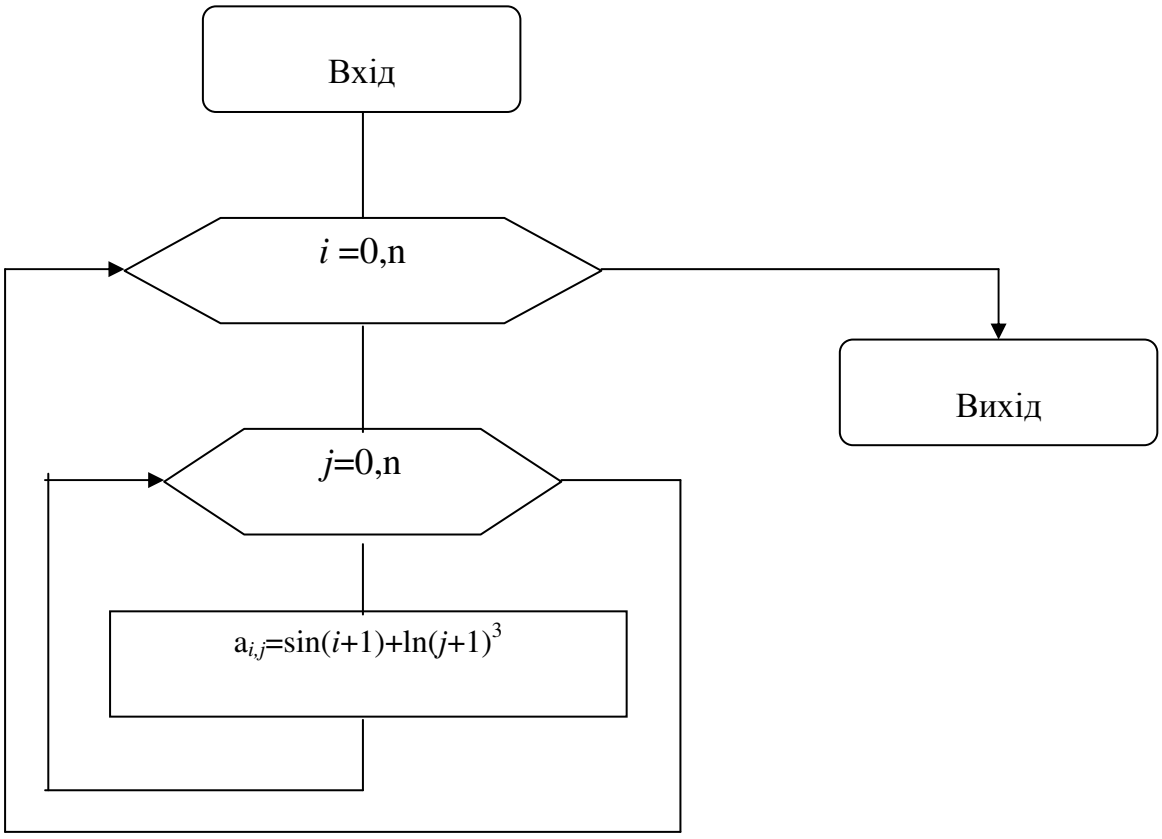

Рисунок 1 – Схема підпрограми обчислення матриці

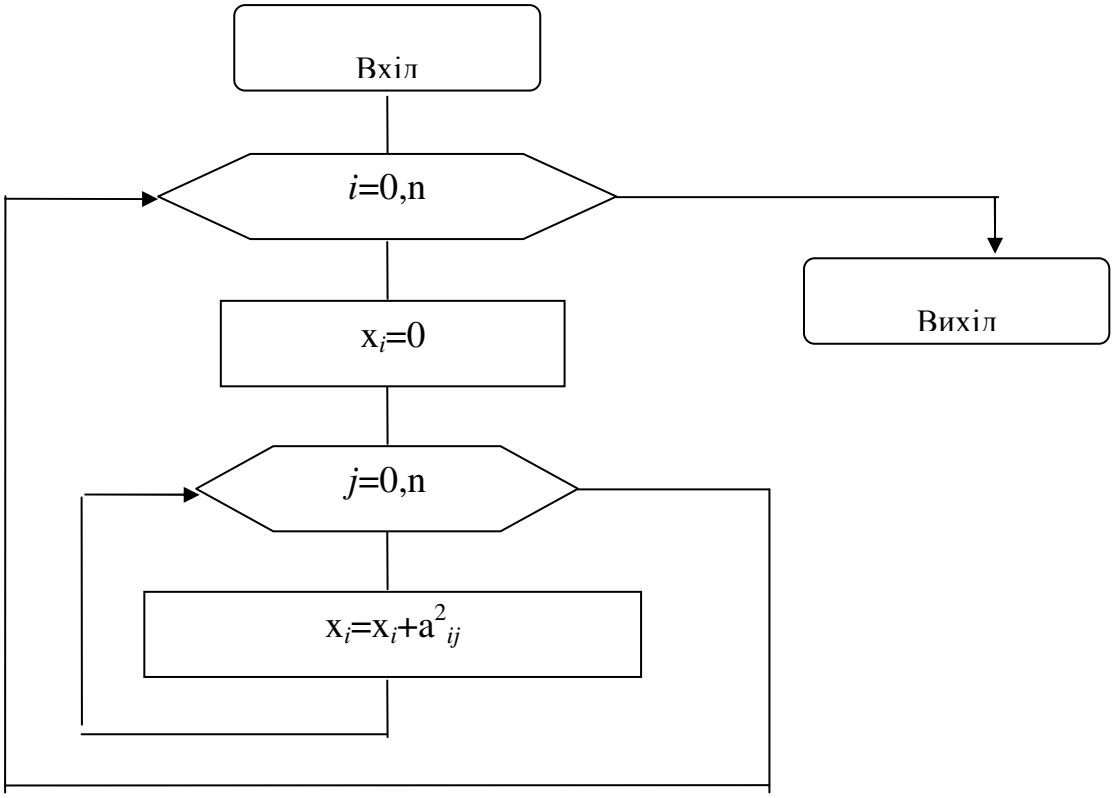

Рисунок 2 – Схема підпрограми обчислення вектора

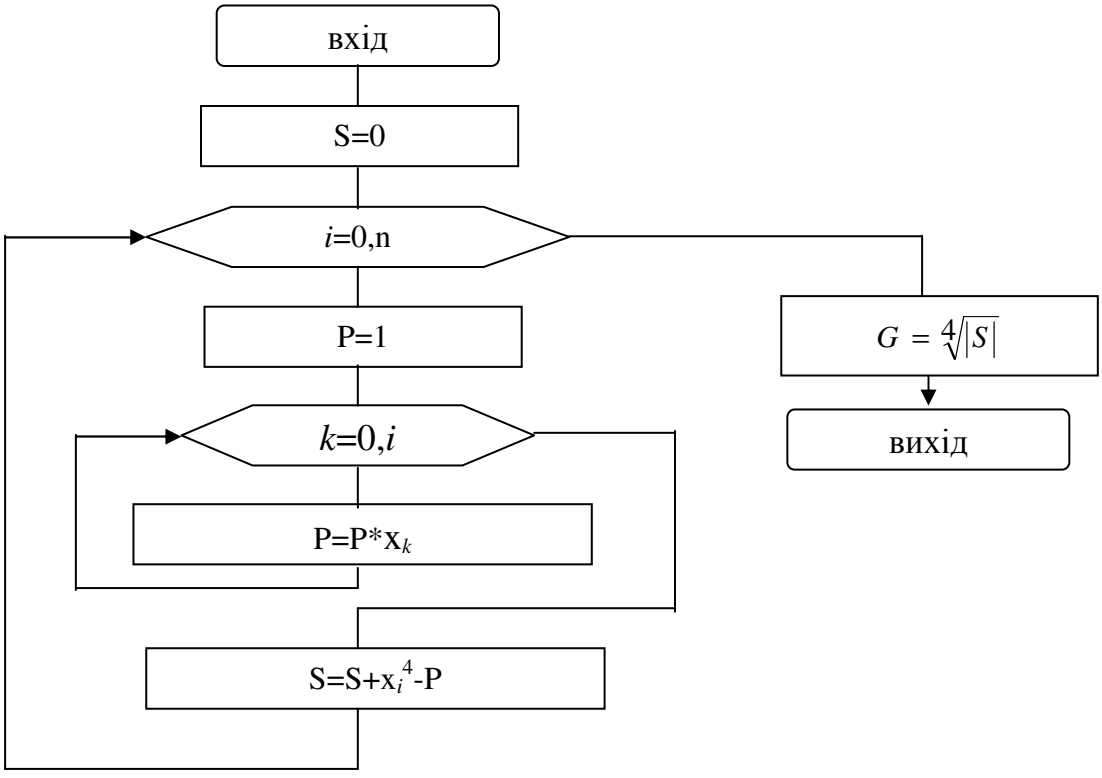

Рисунок 3 – Схема підпрограми обчислення скаляра G

Форма проекту та властивості її компонентів

Розташування компонентів на формі C++Builder-проекту наведено на рис.4.

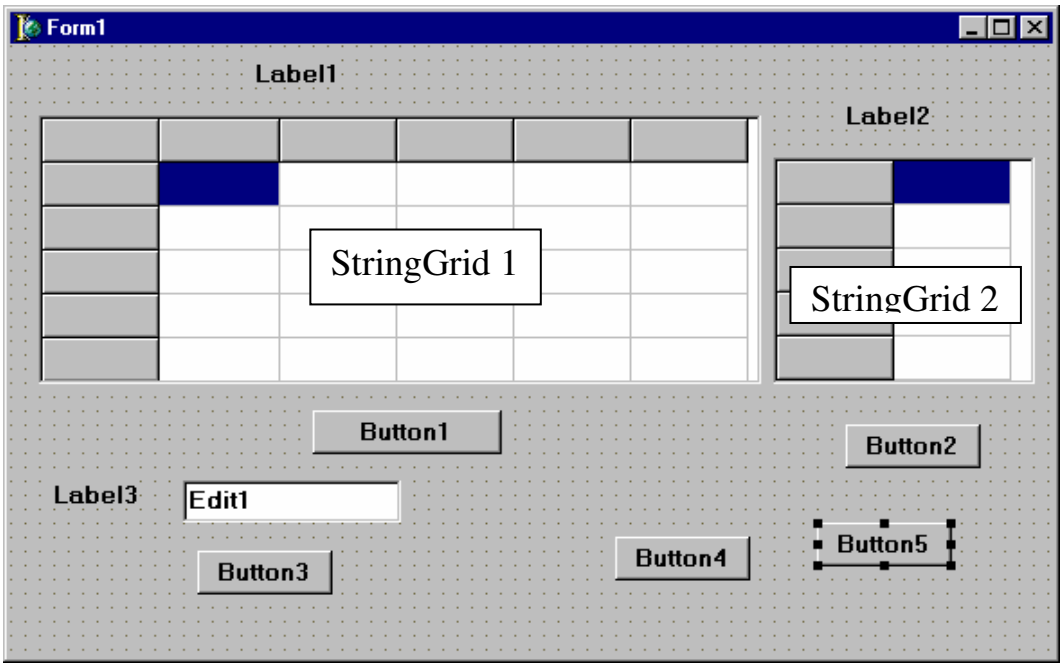

Рисунок 4 – Розташування компонентів на формі

Нестандартні значення властивостей компонентів наведено в табл. 1.

Таблиця 1 – Значення властивостей компонентів на формі

Компонент	**Властивість**	**Значення**
StringGrid 1	ColCount	6
StringGrid 1	RowCount	6
StringGrid 1	FixedCol	1
StringGrid 1	FixedRow	1
StringGrid 2	ColCount	2
StringGrid 2	RowCount	5
StringGrid 2	FixedCol	1
StringGrid 2	FixedRow	0
Label1	Caption	Матриця
Label2	Caption	Вектор
Label3	Caption	Скаляр G
Button1	Caption	Обчислення матриці
Button2	Caption	Обчислення вектора
Button3	Caption	Обчислення скаляра
Button4	Caption	Очистити форму
Button5	Caption	Вихід

Остаточний вигляд форми після зміни значень властивостей згідно табл. 1 наведено на рис.5.

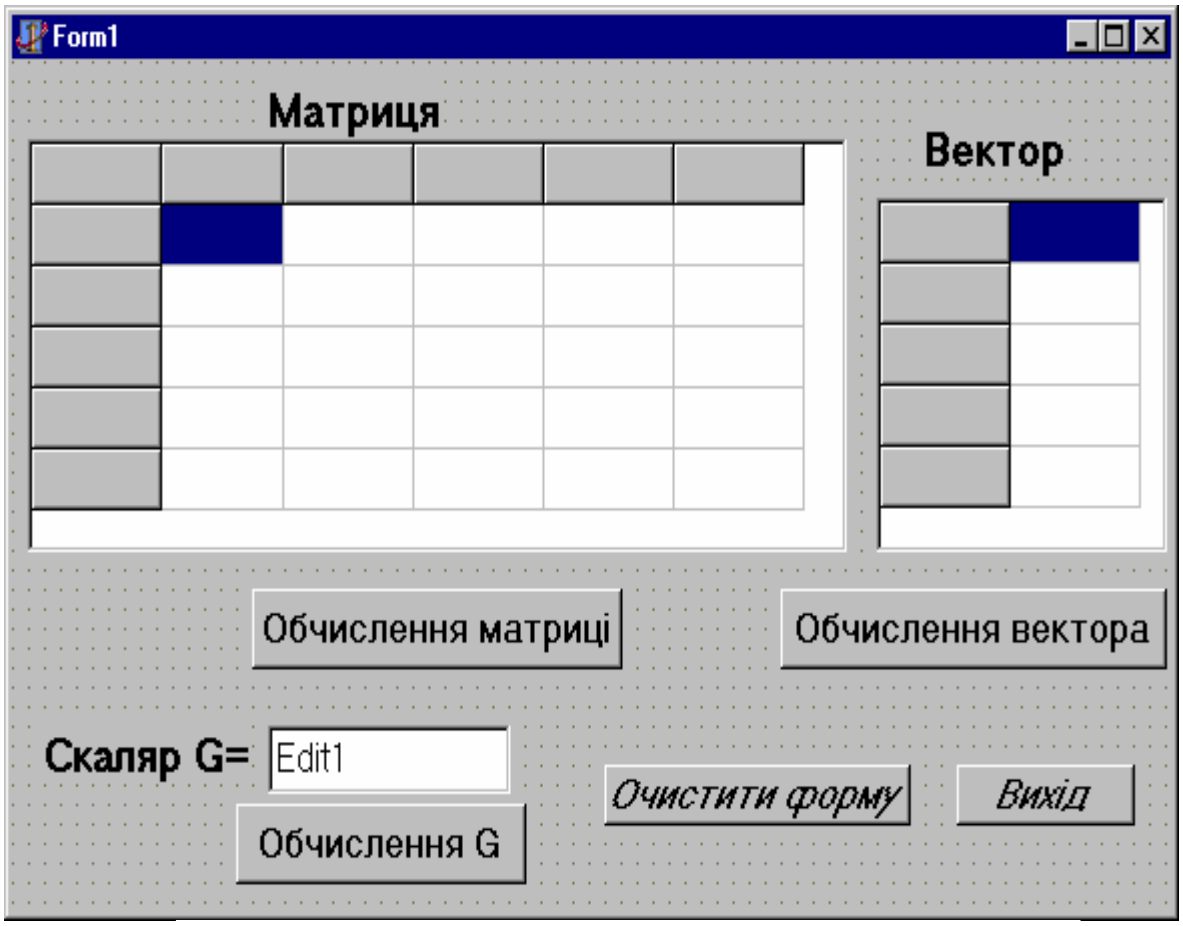

Рисунок 5 – Форма проекту для виконання завдання

Наведемо текст програми:

```
#include <vcl.h>
#pragma hdrstop
// підключимо математичні бібліотеки
#include <math.h>
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------

__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
```

```
// Глобальні параметри
const int n=4;
double a[5][5];
double x[5];
// Підпрограма обчислення елементів матриці

void elem_matr(  )
 {    int i,j;
       for (i=0; i<=n; i++)
       for (j=0; j<=n; j++)
          a[i][j]=sin(i+1)+log(pow(j+1,3));
}
// Підпрограма обчислення вектора сум квадратів елементів рядків

  void elem_vectora (  )
  {int i,j;
      for (i=0; i<=n; i++)
        {
        x[i]=0;
         for (j=0; j<=n; j++)
         x[i]+=pow(a[i][j],2);
        }
  }

 // Підпрограма-функція обчислення скалярної величини G
   double fun_G(  )
         { int i,k; float s,p;
           s=0;
           for (i=0; i<=n; i++)
              {p=1;
                  for (k=0; k<=i; k++)
                       p*=x[k];
                 s+=pow((x[i]),4)-p;
                }
           return pow(fabs(s),1./4);
         }

// Підпрограмма кнопки " Обчислення матриці"
void __fastcall TForm1::Button1Click(TObject *Sender)
  {
    int i,j;
//виклик підпрограми обчислення елементів  матриці
    elem_matr(  );
    for (i=0; i<=n; i++)
```

```
//заголовки рядків
   StringGrid1->Cells[0][i+1]=IntToStr(i)+" -й рядок";
   for (j=0; j<=n; j++)
//заголовки стовпчиків
   StringGrid1->Cells[j+1][0]=IntToStr(j)+" -й стовпчик";
//виведення матриці на  форму
    for (i=0; i<=n; i++)
    for (j=0; j<=n; j++)
    StringGrid1->Cells[j+1][i+1]=FormatFloat("0.000" , a[i][j]);
    }

// Підпрограмма кнопки " Обчислення вектора"

void __fastcall TForm1::Button2Click(TObject *Sender)
{ int i,j;
//виклик підпрограми для вектора
     fun_G(  );
// виведення елементів вектора сум в StringGrid
    for (j=0; j<=n; j++)
      {
       StringGrid2->Cells[0][j]=IntToStr(j)+" -й елемент ";
       StringGrid2->Cells[1][j]=FormatFloat("0.00",x[j]);
       }
}

// Підпрограма кнопки "Обчислення G"
void __fastcall TForm1::Button3Click(TObject *Sender)
{   int i; double G;
//виклик підпрограми-функції обчислення скаляра G
 G=fun_G(  );
//виведення значення G на форму
 Edit1->Text=FormatFloat("0.000",G);
 }

// Підпрограма кнопки " Очистити форму"
void __fastcall TForm1::Button4Click(TObject *Sender)
        { int i,j;
          for (i=0; i<=n; i++)
          for (j=0; j<=n; j++)
            StringGrid1->Cells[j+1][i+1] =" ";
          for (i=0; i<=n; i++)
            StringGrid2->Cells[1][i+1] =" ";
            Edit1->Text=" ";
   }
```

```
// Підпрограма кнопки " Вихід"

void __fastcall TForm1::Button5Click(TObject *Sender)
{
Close( );
}
```

Результати роботи проекту програми

Заключний вигляд форми проекту із результатами наведено на рис. 6.

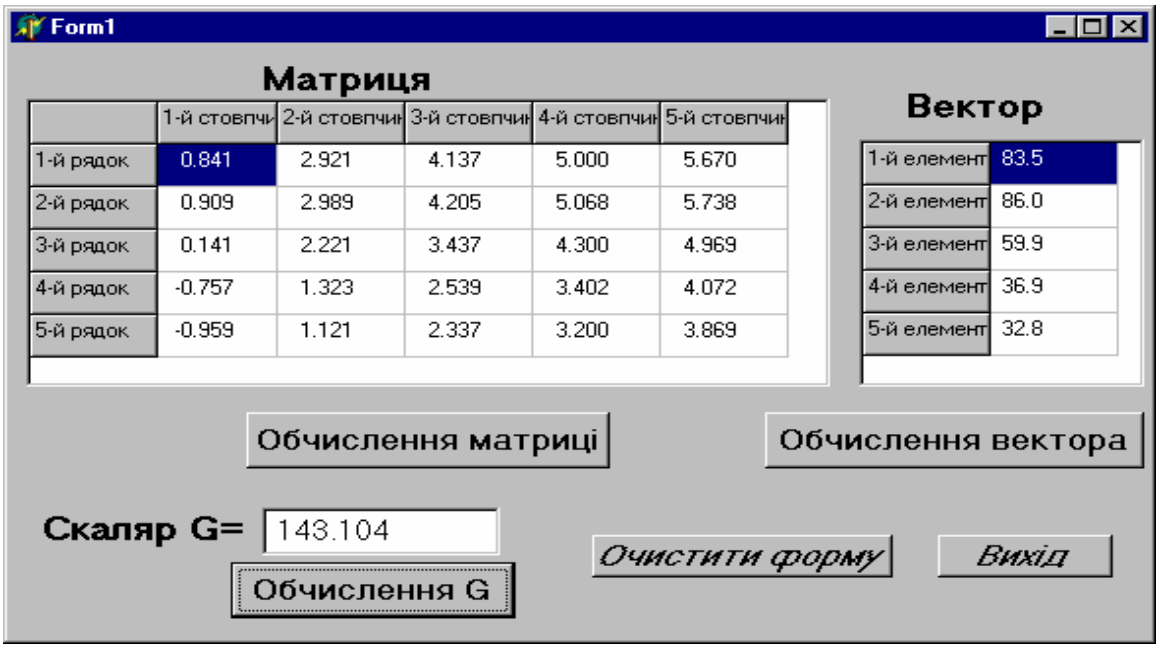

Рисунок 6 – Форма проекту із результатами обчислень

Порядок виконання проекту на комп'ютері

1. Запустимо C++ Builder з робочого столу, клацнувши на піктограмі або скориставшись кнопкою "Пуск" (далі обираємо "Программы"→"C++ Builder ").

2. У вікні форми проекту почнемо проектувати необхідний для роботи програми інтерфейс. Розташуємо на формі необхідні компоненти (наприклад Label, Edit, Button), для введення вхідних даних, перегляду результатів розрахунків та керування роботою проекту. Зразок розташування компонентів у вікні форми наведено на рис. 4.

3. Змінимо значення властивостей компонент за допомогою вікна Object Inspector. Для цього треба виділити необхідну компоненту лівою кнопкою миші та перейти на сторінку Properties вікна Object Inspector. Обрати там властивість CAPTION та набрати нове значення для неї. Приклад значень властивостей компонент наведено у табл.8.

Після виконання цих дій вікно форми буде мати вигляд, зображений на рис. 5

4. Розробимо тексти підпрограм проекту. Тексти підпрограм, які виконуватимуться після клацання по кнопках записують у такий спосіб:

- двічі клацаємо лівою кнопкою миші (подвійне клацання) на кнопці, для якої будемо писати текст програми;
- з'явиться шаблон програми у вигляді :

---

void __fastcall TForm1::Button1Click(TObject *Sender)
{
оператори програми;
}

Аналогічно створимо тексти підпрограм для інших кнопок. Тексти власних підпрограм, які не пов'язані із кнопками, записують перед текстами вищеозначених підпрограм. Для цього треба перейти до вікна із текстом модуля проекту та встановити курсор над першою із підпрограм, що виконуються після клацання на кнопках. Після цього можна набирати підряд тексти власних підпрограм.

Попередження. Не вносьте змін у рядки, які в програмі формує саме C++ Builder !

5. Після запису (або під час запису) текстів підпрограм проекту збережемо створений проект на диску. Для цього оберемо команду меню ***File/SaveAll*** або відповідну піктограму.

У вікні, яке з'явиться, віднайдемо теку, зроблену для цього проекту раніше або створимо нову теку для проекту, скориставшись піктограмою *«Создать папку»* У подальшому після ***Save All*** збереження проекту у необхідній папці здійснюється автоматично.

Зауваження. Бажано кожний проект зберігати в окремій теці.

6. Запустимо проект на перевірку помилок та обчислення, натиснувши клавішу ***F9*** чи  відповідну піктограму, або обравши в меню команду ***Run/Run***.

Коли є помилки, повідомлення про першу з них подається внизу вікна з текстом програми проекту.

Приклади повідомлень:

- *statement missing*; – означає, що очікується символ (**';'**);
- *undefined symbol " X "* – означає, що змінну *x* не описано;
- *call to undefined function* 'cos' – виклик невизначеної функції (тобто не підключена бібліотека математичних функцій);
- *compound statement missing* } – відсутність операторної дужки -};
- *division by zero* – віднайдено ділення на нуль.

Виправивши помилку, знову запускаємо проект на обчислення, після чого з'явиться повідомлення про наступну помилку, якщо вона є. Якщо помилок немає, з'явиться форма виконання проекту (без координатної сітки та

невізуальних компонент). Треба дати команду на збереження проекту без помилок (Save All).

7. Проведемо розрахунки, для чого будемо виконувати необхідні дії (вводити дані, натискати потрібні кнопки, тощо). Зразок форми проекту із результатами наведено на рис. 6.

8. Сеанс завершено, виходимо з C++ Builder .

Література

1. Буката Л.М., Кузнецов В.Д. Інформатика, модуль 1. Ч. 1. - Одесса, ОНАЗ, 2007 – 74с.
2. Буката Л.М., Ясинський В.В., Кузнецов В.Д. Інформатика, модуль 1. Ч. 2. – Одесса: ОНАЗ, 2008 - 104с.
3. Трофименко О. Г., Інформатика, модуль 2.Ч. 1., ч. 2. – Одесса: ОНАЗ: 2008.
4. Прокоп Ю.В. та ін. Інформатика, модуль 3. Ч. 2. – Одесса: ОНАЗ, 2008 - 80с.
5. Буката Л.М., О. Г. Трофименко, В. А. Шаповаленко, К.А.Богатко. Інформатика для студентів заочної форми навчання, Одесса: ОНАЗ, 2009.
6. Леонов Ю.Г., Угрік Л.М., Швайко І.Г. Збірник задач з програмування . – Одесса: УДАЗ, 1997 – 80с.
7. Леонов Ю.Г., Силкина Н.В., Шпинова Е.Д. Программирование инженерных задач: Метод. пособие с элементами лабораторного практикума. – Одесса: ОНАС, 2002. – 68 с.
8. Майкл И., Хаймен. К. Borland C++. Диалектика, 1995 – 416 с.
9. Архангельский А.Я. Програмирование в C++ Builder 5. М.: «Бином», 2000 – 1152 с.
10. Березин Б.Н., Березин С.Б., Начальный курс С и С++. М.: «Диалог-МИФИ», 2000 – 288 с.
11. Калверт Ч., Рейсдорф К. Borland C++, Builder 5.0. Энциклопедия программиста: Пер. с англ.– К.:Издательство “Диасофт” , 2001.- 944с.
12. Бьерн Страуструп. Язык программирования C++. – С.Пб.: М.: Бином, 1999 – 991 с.

Зміст