Міністерство транспорту та зв'язку України
Державна адміністрація зв'язку

ОДЕСЬКА НАЦІОНАЛЬНА АКАДЕМІЯ ЗВ'ЯЗКУ ім. О.С. ПОПОВА

Кафедра інформаційних технологій

Інформатика

МЕТОДИЧНІ ВКАЗІВКИ ТА КОНТРОЛЬНІ ЗАВДАННЯ

для студентів заочної форми навчання

Модуль 1
Організація обчислювальних процесів.
Програмування задач з циклами та масивами

Одеса – 2009

УДК 004.43 План НМВ 2009 р.

Укладачі: Буката Л. М., Трофименко О. Г., Шаповаленко В.А., Богатко К.А.

Методичний посібник містить програму дисципліни “Інформатика”, теоретичні відомості з програмування мовою С++, приклади розв’язання задач та контрольні завдання для складання проектів програм алгоритмічною мовою С++ у середовищі Builder. Наведено вимоги до оформлення контрольного завдання, наведено приклади проектів програм в С++Builder та порядок виконання завдань на комп’ютері.

Методичний посібник призначено для студентів заочної форми навчання усіх спеціальностей академії, які вивчають дисципліну “Інформатика”.

Схвалено:
на засіданні кафедри
Інформаційних технологій
протокол № 2 від 29 вересня 2008 р.

Затверджено:
Методичною радою академії зв'язку
протокол № 6 від 10 лютого 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 проекту в C++Builder.

2. Основи програмування

2.1. Алгоритмізація обчислювальних процессів: 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. Программування 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) елемента масиву. Програмування обчислення кількості елементів згідно заданій умові.

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

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

2.8. Структури.

2.9. Файловий тип: Класифікація даних файлового типу. Текстові файли. Бінарні файли.

3. Обчислювальні методи рішення інженерних задач

3.1. Математичний пакет MathCad: Головне меню. Піктографічне меню. Обчислення за формулами: віразів,сум, добутків, похідних, інтегралів. Табулювання та побудова графіків функції.

3.2. Обчислювальні методи: Обчислення розв'зків нелінійних рівнянь. Обчислення визначених інтегралів. Апроксимація та інтерполяція функцій. Оптимізація функцій.

ВИМОГИ ЩОДО ОФОРМЛЕННЯ КОНТРОЛЬНОЇ РОБОТИ

1. Контрольна робота оформлюється в окремому зошиті чорнилами або пастою. Можна контрольну роботу також набирати на комп'ютері, роздрукувавши її потім на окремих аркушах. Сторінки роботи повинні бути перенумерованими.
2. Слід заповнювати тільки одну сторону аркуша з залишенням полів для зауважень викладача.
3. Для **кожної задачі** контрольної 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.

Елементи мови С++

Алфавіт мови – набір символів, які можна використовувати для запису команд у програмі, – це великі та малі латинські букви, цифри та спеціальні знаки.

Змінна – іменована ділянка оперативної пам'яті. Застосовується для зберігання даних під час роботи програми.

Значення змінної – фактичне значення, що знаходиться в цій області пам'яті. Тип змінної – вид даних, які змінна може зберігати. Кожний тип змінних зберігається й обробляється за певними правилами. Тип змінної ще визначає, скільки пам'яті займе змінна. У програмі мовою С++ усі змінні повинні бути визначені, тобто для кожної змінної повинен бути вказаний її тип.

Основні типи С++

Дані цілого типу визначають множину цілих чисел у різних діапазонах. Цілі типи позначаються ідентифікаторами **int**, **unsigned int**, їхні характеристики наведено у табл.1.

Дані дійсного типу – це дані, котрі в якості своїх значень приймають числа з фіксованою чи плаваючою крапкою.

Дійсні типи позначаються ідентифікаторами **float** та **double** та мають характеристики, наведені у табл. 1.

Дані символьного типу - це будь-який символ мови, взятий в апострофи, наприклад '–', 'A', '7'.

Змінна символьного типу (char) – це змінна, яка набирає значення символьної константи. Усі типи мови С++ розподіляють на дві групи: основні типи та структуровані. До основних можна віднести **char**, **int, float** та **double**, а також їх варіанти з модіфікаторами **short** (короткий), **long** (довгий), **signed** (зі знаком) та **unsigned** (без знака).

Наприклад:

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

Визначати змінні можна у будь-якому місті програми, але перед їх використанням.

Структуровані типи включають у себе вказівники, масиви будь-яких типів, типи функцій, класи, структури.

Таблиця 1 ─ Діапазон значень типів даних

Тип	**Назва**	**Розмір** (в байтах)	**Діапазон**	**Приклади можливих значень**	**Типи чисел**
char	символьний (знаковий)	1	–128…126	'a', '\n', '9'	цілі
unsigned char	беззнаковий символьний	1	0…255	1, 233	
short	короткий цілий	2	–32 768…32 767	1, 153, –349	
unsigned short	беззнаковий короткий	2	0…65 535	0, 4, 65 000	
int	цілий (знаковий)	4	–2 147 483 648 …2 147 483 647	–30 000, 0, 690	
unsigned int	беззнаковий цілий	4	0…4 294 967 295		
float	дійсний одинарної точності	4	$3.4\cdot10^{-38}$…$3.4\cdot10^{38}$	3.23, 12312, –0.947, 0.0001, 6.34e–3, 4e5	дійсні
double	дійсний подвійної точності	8	$1.7\cdot10^{-308}$ …$1.7\cdot10^{308}$		
long double	довгий дійсний	10	$3.4\cdot10^{-4932}$ …$1.1\cdot10^{4932}$		
bool	логічний	1	false, true	false(0), true(>=1)	
AnsiString, String	рядок			”Hello!”	
void	порожній, без значення	–	–	–	
enum	перераховний	4	–2 147 483 648 …2 147 483 647		

Константи у мові С++

Константа – величина, яка не змінюється впродовж виконання програми. Константи можуть бути любого типу, наприклад:

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

У мові С++ використовуються рядкові константи. Рядкові константи або рядок представляють собою набір символів у подвійних лапках, наприклад “Програмування”. Для оголошення константи у програмі використовується модифікатор **const**. При цьому вказується тип константи. Наприклад:

const float Pi=3.14159;.

У якості значення константи можна вказувати константний вираз, який містить оголошенні константи. Наприклад:

const float Pi2=2*Pi;

const float K=Pi/180;.

Для цілих констант тип можна не вказувати.

Операції мови С++

Таблиця 2 - Арифметичні операції

Позначення	**Операція**	**Приклад**
+	Додавання	X+Y
-	Віднімання та унарний мінус	X-Y
*	добуток	X*Y
/	Ділення	X / Y
%	Остача від цілочисельного ділення	I % 6
++	Зростання на одиницю (інкремент)	i++ ++i
--	Зменшення на одиницю (декремент)	i- - - -i

Операції додавання, віднімання, множення та ділення виконуються зліва направо, тобто спочатку обчислюється значення лівого операнда, а потім значення правого. Якщо операнди мають один і той же тип, то результат буде мати той же тип. Тому, якщо операція ділення використовується до цілих змінних, то результат буде теж цілим.

У мові С++ використовуються ще дві операції, яких нема в інших мовах. Це унарні операції ++ (інкремент) та - - (декремент). Унарною називається операція, яка має тільки один операнд. Операція ++ добавляє одиницю до операнда, операція - - віднімає одиницю з операнда. Обидві операції можуть стояти перед операндом (префіксна форма запису) та після операнда (постфіксна форма запису). При префіксній формі змінна спочатку збільшується або зменшується на одиницю, а потім це її нове значення використовується в тому виразі, в якому вона зустрілась. При постфіксній формі у виразі використовується поточне значення змінної і тільки тоді змінна збільшується або зменшується на одиницю.

Оператор присвоєння

Оператор присвоєння – основний оператор мови програмування – має такий формат: ***ім'я змінної = вираз*;**

Цей оператор обчислює значення виразу і надає здобуте значення змінній, при цьому тип виразу повинен відповідати типу змінної. Припускають надання змінній дійсного типу значення виразу цілого типу.

Надання змінній цілого типу виразу дійсного типу *заборонено!*

Крім простого присвоювання всі інші є складними операціями. Вони присвоюють результат змінній зліва, вказаній перед символом «=».

Наприклад, вираз **X+=Y** еквівалентний виразу **X=X+Y**, але записується компактніше та виконується швидше.

Арифметичні вирази

Арифметичні вирази будуються з арифметичних констант, змінних, функцій і операцій над ними. Обчислення виконуються зліва направо у відповідності з таким старшинством операцій:

1. Стандартні функції, ++, --.
2. Множення (*), ділення (/), остача від ділення (%).
3. Додавання (+) й віднімання (–).

Вирази в круглих дужках виконуються в першу чергу.

Приклади запису арифметичних виразів надані у табл. 3.

Таблиця 3 – Запис виразів мовою С++

Математичний запис виразу	**Запис виразу мовою С++**
$y = \cos x^2 + \sin^2 x$	y=cos(x* x)+ pow(sin(x),2);
$y = \sin^4 x$	y=pow(sin(x), 4);
$y = \sqrt[5]{x}$ (для програмування використовують заміну $\sqrt[5]{x} = x^{1/5}$)	y=pow(x, 1./5);
$y = \frac{\sin(x-\pi)+1}{e^x - 2.5x}$	y=(sin(x-M_PI)+1)/(exp(x)-2.5*x);

Стандартні функції перетворення типів даних

Введення й виведення даних в С++ відбувається за допомогою змінних рядкового типу. Якщо написати, наприклад в Edit1 – 3.25, то це не число, а його зображення у вигляді рядка символів. З ним жодних обчислень робити не можна до перетворення типу string на числовий тип, а потім навпаки. В мові С++ це здійснюють стандартні підпрограми. Наведемо приклади їхнього застосування :

- X=StrToFloat(S); //функція, що перетворює string-рядок S на дійсне число X;
- X=StrToInt(S); // функція, що перетворює string-рядок S на ціле число X;
- S=FloatToStr(X); // функція, що перетворює дійсне число X на string-рядок S;
- S=IntToStr(X); //функція, що перетворює ціле число X на string-рядок S;

Основні математичні функції

Функція	Опис	Бібліотека
int **abs**(int i)	модуль цілого числа **x**	stdlib.h
double **ceil**(double x)	округлення вверх: найменше ціле, не менше за **x**	math.h
double **exp**(double x)	експонента	math.h
double **fabs**(double x)	модуль дійсного числа **x**	math.h
double **floor**(double x)	округлення вниз: найбільше ціле, не більше за **x**	math.h
Extended **IntPower**(Extended Base, int Exponent)	піднести **Base** до цілого степеня **Exponent**	Math.hpp
double **log**(double x)	натуральний логарифм	math.h
double **log10** (double x)	десятковий логарифм	math.h
Extended **LogN**(Extended Base, Extended X)	логарифм **X** по основанню **Base**	Math.hpp
double **pow**(double x, double y)	$\mathbf{x}^{\mathbf{y}}$	math.h
double **sqrt**(double x)	корінь квадратний	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

Середовище програмування C++ Builder

C++ Builder – це технологія візуального програмування, яка дає змогу замість повного самостійного написання програми використовувати великий набір готових візуальних об'єктів, так званих *компонентів*. Піктограми компонентів розміщено на відповідних вкладках палітри компонентів. Компоненти забезпечено набором певних *властивостей* (назва, колір, розмір тощо), *методів* (програмних кодів, готових до виконання) та *шаблонів підпрограм-відгуків на обробку подій*

(підпрограм-оброблювачів). Ці події (натиснення клавіші миші, вибір пункту меню тощо) виникають під час виконання програми. Найбільш поширеною є подія OnClick – натиснення лівої кнопки миші на об'єкті.

Властивості компонентів обираються на сторінці *Properties*, а підпрограми-оброблювачі – на сторінці *Events* у вікні *Object Inspector*. "Контейнером", в якому на екрані розміщуються компоненти, слугує *форма*, яка сама є компонентом з назвою Form. Без додаткових вказівок заголовок компонента збігається з його назвою, до якої додається порядковий номер починаючи з 1. Але назву можна змінити за допомогою властивості *Caption* .

Базові компоненти

Компоненти, які найчастіше використовуються, містяться на вкладці *Standard* палітри компонент. Перелічимо деякі з них.

Label – надпис, який слугує для написання пояснювального тексту на формі з використанням властивостей Caption і Font ;

Button– кнопка призначена для відгуку на подію;

Edit– однорядковий редактор, який слугує для введення та виведення лише одного рядка тексту; цей рядок зберігається у властивості Text.

Як тількі на формі з'являється компонент, Object Inspector відображає інформацію про нього.

Вікно C++ Builder представлено на рис. 1.

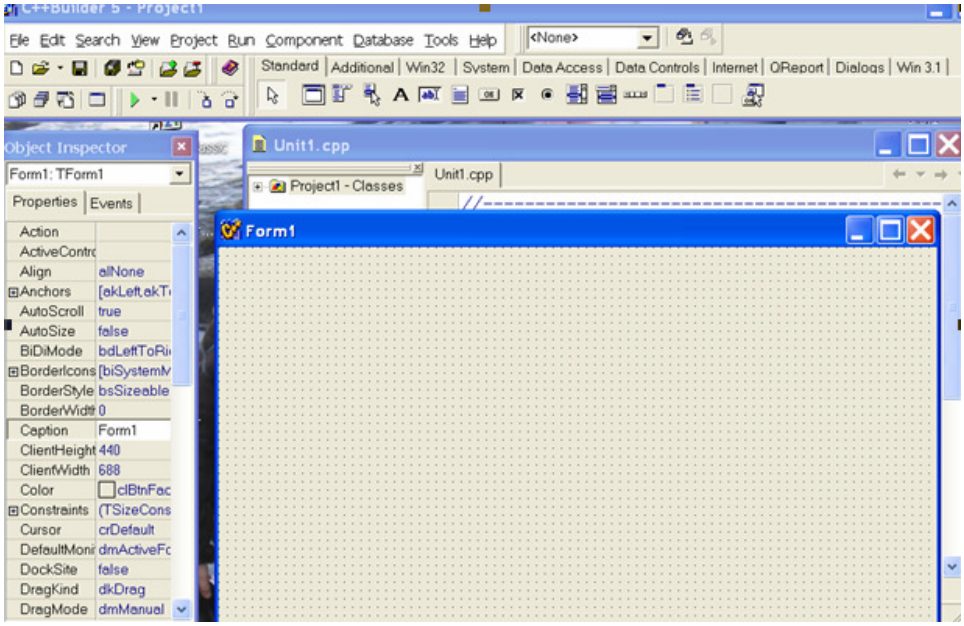


Рисунок 1– Головне вікно C++ Builder

Послідовність роботи у C++ Builder

Послідовність роботи у **C++ Builder** може бути такою:

– винести на форму з палітри компонент всі необхідні компоненти для створення діалогового вікна програми;

– задати на вкладці **Properties** (властивості) ***Object Inspector*** всі необхідні властивості для компонент, розташованих на формі;

– створити необхідні відгуки на події для розташованих на формі компонент, за допомогою вкладки **Events** (події) ***Object Inspector*** або безпосередньо з форми, клацнувши двічі по компоненту;

– у вікні Редактора Кодів заповнити відгуки на події кодом, а також написати текст всіх необхідних програм. Перехід з вікна форми на вікно редактора здійснюється клавішею **<F12>**;

– зберегти створений проект за допомогою команди головного меню **File/Save Project As**;

– запустити проект на виконання одним із способів: натиснути клавішу **<F9>**, виконати команду меню **Run/Run**, натиснути відповідну кнопку на панелі інструментів;

– якщо помилок немає, з'явиться форма без координатної сітки. Якщо помилки є, треба їх виправити і знову запустити проект на обчислення. Треба дати команду на збереження проекту без помилок (Save All);

– проведемо розрахунки, для чого будемо виконувати необхідні дії (вводити дані, натискати потрібні кнопки, тощо);

– виходимо з ***C++ Builder***. Сеанс закінчено.

Організація проекту у C++ Builder

Проект C++ Builder складається з декількох файлів: файлів форм, модулів, ресурсів тощо. Головною частиною проекту у C++ Builder є головний файл проекту (**.cpp**) з функцією **WinMain**, з якої починається виконання програми і яка забезпечує ініціалізацію інших модулів. Він створюється автоматично і не треба його змінювати, цей файл має ім'я **Project1**.

Основний файл, з яким ви працюєте – файл реалізації модуля (**.cpp**). В ньому зберігається код, відповідний до форми.

У заголовочному файлі з розширенням **.h** зберігається опис класу цієї форми. Весь основний текст цього файла формується автоматично, але іноді треба вводити в нього описи своїх функцій, типів, змінних. Відкрити цей файл в редакторі кодів можна клацнувши у вікні з файлом реалізації модуля правою кнопкою миші та вибравши з локального меню команду **Open Sourse/ Header File.**

Імена заголовочного файла та файла реалізації співпадають. Ви задаєте це ім'я, коли в перший раз зберігаєте проект. За замовчуванням C++ Builder пропонує ім'я **Unit1.**

Структура програми на С++

Програма на С++ складається з оголошень (змінних, констант, типів, класів, функцій) та опису функцій. Серед функцій завжди є головна – **main** для консольних додатків (яка працює з WIN32) або **WinMain** для додатків Windows – ця головна функція виконується після початку роботи програми. Функцію **WinMain** містить головний файл проекту, який створюється автоматично і його змінюють тільки в виключних випадках. Щоб побачити текст головного файла, треба виконати команду **Project/View Source.**

Програми на С++ створюються за модульним принципом і складаються з багатьох модулів. Всі об'єкти компонентів розташовуються на формах. Для кожної форми C++ Builder створює окремий модуль.

Заголовочний файл має вигляд:

```
//---------------------------------------------------------------------------
#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
// тут можуть розміщуватися додаткові директиви
// препроцесора (зокрема, include),
// які не включаються у файл автоматично
//---------------------------------------------------------------------------
// об'явлення класу форми TForm1
class TForm1 : public TForm
{
__published:        // IDE-managed Components
// розташовані  на формі компоненти
        TLabel *Label1;
        TButton *Button1;
        void __fastcall Button1Click(TObject *Sender);

private:        // User declarations
// закритий розділ класу
// тут можуть розміщуватися оголошення типів, змінних, функцій,
// які входять в клас форми, але не доступних для інших модулів
public:                 // User declarations
```

```
// відкритий  розділ класу
// тут можуть розміщуватися оголошення типів, змінних, функцій,
// які входять в клас форми, доступних для інших модулів

__fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
// тут можуть розміщуватися оголошення типів, змінних, функцій,
// котрі не включаються в клас форми;
// доступ до них з інших блоків тільки при дотриманні
// деяких додаткових умов
#endif
```

Починається заголовочний файл рядками, перший символ яких – #. З цього символу починаються директиви препроцесора. Серед них найбільш важливі директиви **include**. Ці директиви підключають у модуль вказані в них файли і, зокрема, в заголовочному файлі розміщені директиви **include**, які підключають копії файлів, в котрих знаходиться опис компонентів, змінних, функцій, що використовуються у даному модулі. Але, для деяких функцій таке автоматичне підключення не робиться. В таких випадках, якщо ви хочете у своїй програмі використовувати математичні функції, необхідно підключити бібліотеку, яка містить ці функції, тобто власноруч ввести директиви:

```
#include <math.h>     // підключення бібліотеки math.h
#include <Math.hpp>  // підключення бібліотеки Math.hpp
```

Після директив препроцесора йде опис класа форми – **TForm1**. Розділ **__published** заповнюється автоматично в процесі проектування форми. В даному прикладі ви бачите оголошення покажчиків на два компонента: **Label1** типу **TLabel** та **Button1** типу **TButton**. Там є об'явлення функції **Button1Click** – оброблювача події на натискання кнопки **Button1**. Розділи **private** та **public** заповнюються розроблювачем програм. Те, що оголошено у розділі **public**, буде доступно для інших класів і модулів. Те, що оголошено у розділі **private**, доступно тільки у межах даного модуля.

Нижче розділу **PACKAGE** можна розмістити оголошення типів, змінних, функцій, до яких буде доступ з інших модулів, але які не включаються в клас форми.

Тепер розглянемо текст файла реалізації.

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
```

```
#pragma resource "*.dfm"
// тут можуть розміщуватися додаткові директиви
// препроцесору (зокрема, include),
// які не включаються у файл автоматично
// оголошення класа форми TForm1
TForm1 *Form1;
//---------------------------------------------------------------------------
// виклик конструктора форми Form1

__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
// тут можна розмістити оператори,
// які повинні виконуватися при  створенні форми
}

// тут можуть розміщуватися оголошення типів, змінних,
// доступ до яких з інших модулів можливий тільки при
// виконанні деяких додаткових умов;
// тут повинні бути  реалізації  всіх функцій, які оголошені в
// заголовочному файлі, а також  можуть бути реалізації  будь-яких
// додаткових функцій, не оголошених раніше
```

Програми з лінійною структурою

Задача 1.

Скласти схему алгоритму та проект роботи програми у середовищі програмування C++Builder для обчислення функції

$$y = x^2 + \sqrt[3]{|x|};$$

$$x = \cos^2(b) + \sin^2(a);$$

$$a = \sqrt{b + t^2},$$

де значення b=7,1 – задати як константу, а значення t = 2 ввести за допомогою компонента Edit.

Структурну схему розв'язання задачі наведено на рис.2. На рис. 3 наведено вікна форм проекту задачі в C++: на етапі створення та із результатами обчислень.

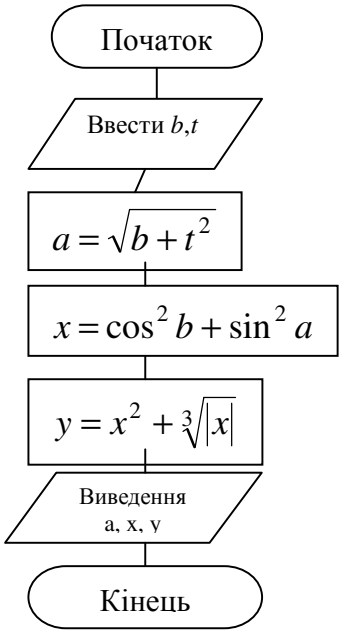


Рисунок 2 – Структурна схема до задачі 1

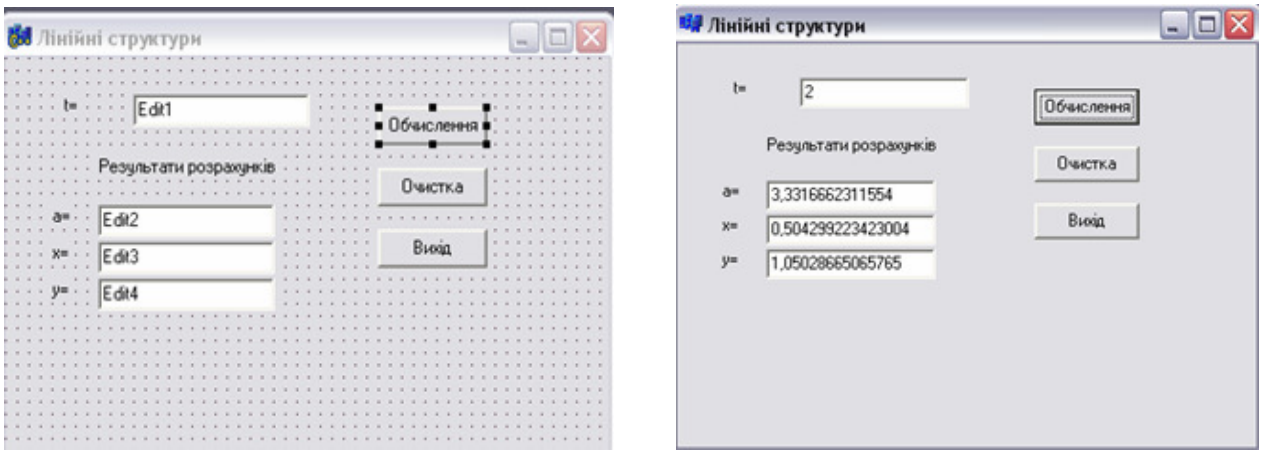


Рисунок 3 – Вікна форм до задачі 1

Текст програми:

```
#include <math.h> //підключення бібліотеки математичних функцій
        …
  // текст програми для кнопки "Вихід"

void __fastcall TForm1::Button3Click(TObject *Sender)
{
Close();
}
```

```
//---------------------------------------------------------------------------
        // текст програми для кнопки "Очистка"
void __fastcall TForm1::Button2Click(TObject *Sender)
{ Edit1->Clear(); //очищення компонента Edit1
  Edit2->Clear();//очищення компонента Edit2
  Edit3->Clear();//очищення компонента Edit3
  Edit4->Clear();//очищення компонента Edit4
}
//---------------------------------------------------------------------------
     // текст програми для кнопки "Обчислення"

void __fastcall TForm1::Button1Click(TObject *Sender)
{
 const float b=7.1; //опис константи дійсного типу
  float t,a,x,y;
  t=StrToFloat(Edit1->Text); //введення значення t
  //обчислення за заданими формулами
  a=sqrt(b+pow(t,2));
  x=pow(cos(b),2)+pow(sin(a),2);
  y=pow(x,2)+pow(fabs(x),1./3.);
    // виведення результатів обчислень
   Edit2->Text=FloatToStr(a);
   Edit3->Text=FloatToStr(x);
   Edit4->Text=FloatToStr(y);
}
```

Алгоритмічні структури розгалуження

Розгалуженим називається обчислювальний процес, якщо, він виконується за однією з певних, заздалегідь передбачених, гілок алгоритму. На блок-схемі (рис. 2) структури розгалуження позначаються ромбами.

Для програмної реалізації таких обчислень використовують оператори передачі управління, котрі дозволяють змінювати порядок виконання операторів програми. У мові С++ для цього передбачені оператор безумовного переходу – **goto**, оператор умовного переходу – **if** та оператор множинного вибору варіантів – **switch**.

Оператор безумовного переходу goto

Загальна форма опису оператора **goto** має вигляд: **goto** <мітка >;

Оператор **goto** (йти до) дозволяє передати керування в будь-яку точку коду (програми), котру позначено спеціальною міткою.

Мітки, на які можна передавати керування, у мові С++ не описують. Кожна мітка може позначатися припустимим в алгоритмічній мові ідентифікатором, який обов'язково починається з латинської літери. Наприклад :

start, M55, a;

означає три мітки: start, M55, a. В С++ оператор **goto** дуже рідко застосовується.

Умовний оператор

Умовний оператор у С++ дозволяє виконувати один або два оператори, які входять до нього, залежно від значення логічного виразу.

Оператор **if** має дві форми: скорочену та повну. Скорочена форма має вигляд:

if (логічний вираз) оператор;

Повна форма цього оператора наступна:

if (логічний вираз) оператор1; else оператор2;

де **if** (якщо), **else** (інакше) – службові слова; оператор1, оператор2 - прості чи складені оператори мови.

Порядок виконання умовного оператора наведено на рис.4.

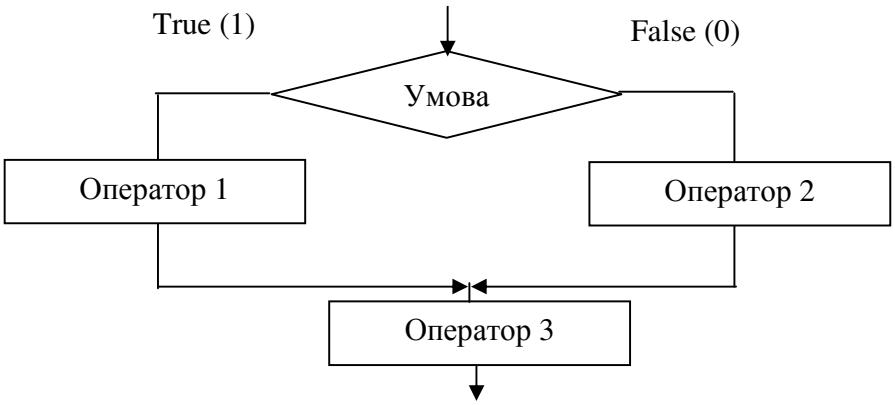


Рисунок 4 – Схема алгоритму умовного оператора переходу

Як видно із схеми, якщо значення логічного виразу дорівнює true, то виконується оператор1, якщо логічний вираз – false, то виконується оператор2 (оператор 1 пропускається). Далі кожного разу виконується оператор 3, що стоїть за оператором **if**. Наприклад, фрагмент програми обчислення функції

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

має вигляд : **if** (*x*>0) *y*=ln(*x*); **else** *y*=exp(*x*);

Тут логічний вираз – відношення x>0, оператор 1 і оператор 2 – оператори присвоєння *y*=ln(*x*) і *y*=exp(*x*).

Порядок виконання оператора скороченої конструкції if (без else) показано на рис. 5. Якщо логічний вираз набирає значення False, то виконується наступний за **if** оператор.

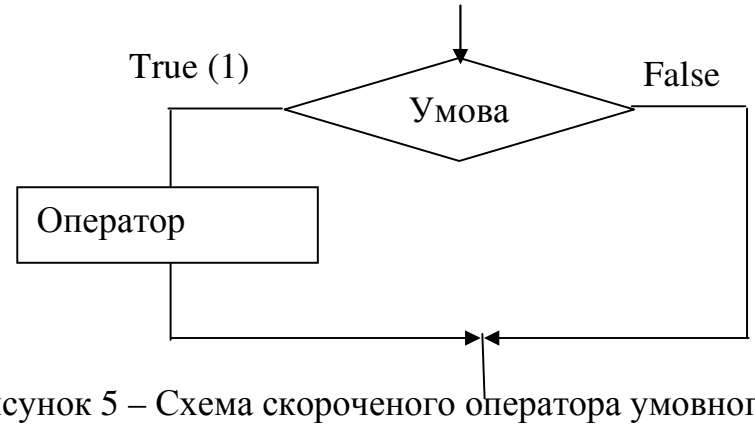


Рисунок 5 – Схема скороченого оператора умовного переходу

Приклад 1. Ввести значення координат *(x, y)* точки площини. Визначити, якій чверті площини належить ця точка.

……

```
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 лежить на вісі");
  } else
  if ((x>0)&&(y>0))  nomer=1;
    else  if ((x<0)&&(y>0)) nomer=2;
      else  if ((x<0)&&(y<0))  nomer=3;
                else  nomer=4;
        Edit3->Text=IntToStr(nomer(x,y));
    }
```

Функція ShowMessage("**Повідомлення**"); – призначена для виводу на екран діалогового вікна з текстом повідомлення з однією командною кнопкою ОК.

Циклічні обчислювальні процеси

Обчислювальний процес називається *циклічним*, якщо він неодноразово повторюється допоки виконується деяка задана умова. Блок повторюваних операторів називають *тілом* циклу. Оператори циклу можуть бути таких типів:

– оператор циклу for;
– оператор циклу з передумовою while;
– оператор циклу з післяумовою do-while.

Оператор циклу з параметром for

Як правило, цей оператор використовується, коли заздалегідь відома кількість повторювань. Прикладами використання даного оператора можуть бути обчислення сум заданої кількості елементів масиву, пошук мінімального (максимального) елемента масиву, сортування елементів масиву за збільшенням (за зменшенням) тощо.

Синтаксис оператора:

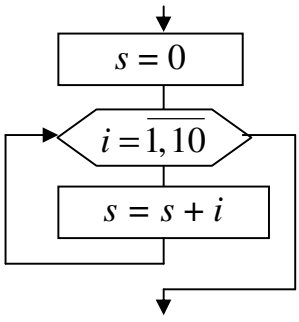
for (*вираз1; вираз2; вираз3*) *оператор*;

Вираз1 задає початкове значення змінних (параметрів), які керують циклом, потім перевіряється умова в *виразі2,* яка є умовою продовження виконання циклу. Якщо умова виконується (істина, має ненульове значення), то виконується *оператор* (чи група операторів в операторних дужках {}), після чого *вираз3* змінює параметри циклу, і, у випадку істинності умови, виконання циклу продовжується. Якщо *вираз2* дорівнює нулю (умова не виконується), то керування передається на оператор, наступний за оператором for.

Суттєво те, що перевірка умови виконується на початку циклу. Це означає, що тіло циклу може не виконуватись жодного разу, якщо умова на початку неправдива.

Простий приклад для обчислення суми $S = \sum_{i=1}^{10} i$ проілюструє використання оператора for:

```
int   s = 0;
for (int i = 1; i <= 10; i++)  s += i;
```



Можлива наявність пустого оператора (відсутній оператор) у тілі циклу. Так, суму із попереднього прикладу можна обчислити інакше:

```
for (int s = 0 ,  i = 1;  i <= 10;  s += i++);
```

У цьому прикладі *вираз1* включає в себе два оператори, розділених операцією кома, які задають початкові значення змінних s та i.

У другому варіанті рішення відсутній *оператор*, а на початку задається значення двох змінних. В операторі можливі конструкції, коли відсутній той чи інший вираз: *вираз1* може бути відсутній, якщо початкове значення задати до цього;

вираз2 – якщо припускається, що умова завжди істинна, тобто слід обов'язково виконувати тіло циклу допоки не зустрінеться break; а *вираз3* – якщо приріст параметра здійснювати в тілі циклу або взагалі непотрібно. Тоді сам вираз пропускається, але точка з комою обов'язково повинна бути.

Приклади оператора циклу для обчислення факторіалу F=n! (нагадаємо, що факторіал обчислюється за формулою (n)!=1·2·3·...·(n–2)·(n–1)·n. Наприклад, 4!=1·2·3·4=24):

1) int F = 1, n = 5; for (int i = 1; i <= n; i++) F *= i;
2) int F, i, n = 5; for (F = 1, i = 1; i <= n; F *= i++);

Розглянемо декілька прикладів розв'язання задач, в яких доцільно використання оператора циклу for. Для усіх задач розроблені алгоритмічні схеми (блок-схеми), які визначають порядок виконання дій, і приведені форми із результатами обчислень.

Загальним правилом обчислення сум кінцевої чи нескінченої кількості доданків є необхідність початкового обнуління змінної, в якій будуть підсумовуватися усі доданки (див. задачу 2). Для обчислення добутку (див. задачу 1.2) перед циклом необхідно змінній, в якій будуть перемножуватись усі множники, присвоїти початкове значення 1. Ці дії необхідні, оскільки при першому виконанні циклу додавання нуля (чи множення на одиницю) не викривить значення результату.

Задача 2. Обчислити суму кінцевого числа доданків $S = \sum_{i=-3}^{3} \frac{(i+2)(i+4)}{(i-2)(i+5)}$, де $i = -3, -2, \ldots, 2, 3$, використовуючи в програмі оператор циклу з параметром.

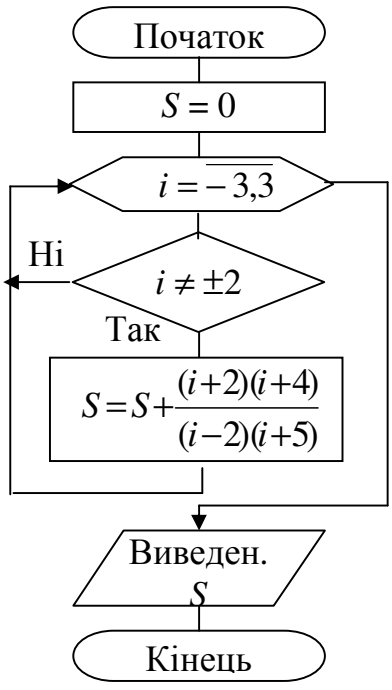


Рисунок 6 – Блок-схема до задачі 2

При виконанні обчислень не слід враховувати ті доданки, в яких чисельник або знаменник дорівнюють нулю. В даному прикладі при значенні $i = -2$ утворюється нуль в чисельнику, а при $i = 2$ – в знаменнику, тому умовний оператор if допоможе виключити ці доданки.

Схема алгоритму програми та приклад форми із результатами роботи наведені на рис. 6 і 7.

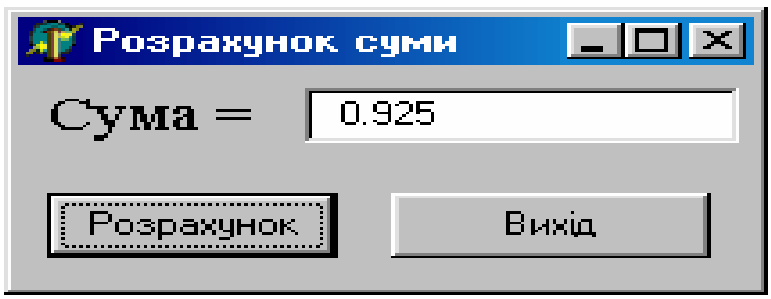


Рисунок 7 – Форма з результатами роботи

Текст програми:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
  {  int i;  float S = 0;
     for (i = -3; i <= 3; i++)
       if ( i != 2  &&  i != -2)   S += (float) (i + 2) * (i + 4) / ((i - 2) * (i + 5));
    Edit1->Text = FormatFloat("0.000", S); }
```

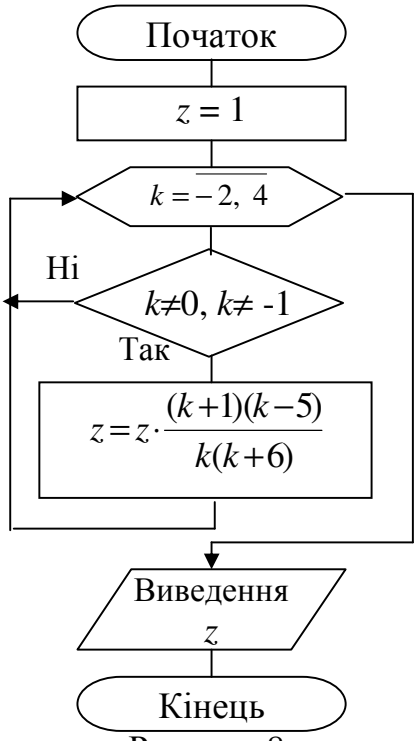


Рисунок 8 – Блок-схема до задачі 3

Задача 3. Обчислити добуток $z = \prod_{k=-2}^{4} \frac{(k+1)(k-5)}{k(k+6)}$, де $k = -2, -1, \ldots, 3, 4$.

Обмеження такі ж, як і в попередній задачі. Схема алгоритму програми і приклад форми із результатами роботи наведені на рис. 8 і 9.

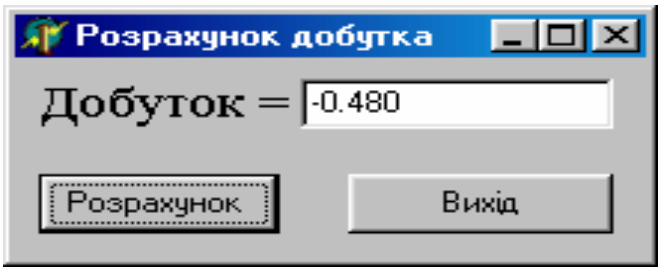


Рисунок 9 – Форма із результатами роботи

Текст програми:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{    int k;  float p, z = 1;
     for (k = -2; k <= 4; k++)
if (k != -1 && k != 0) z *= (float) (k + 1)*(k- 5)/(k * (k+6));
Edit1->Text = "z=" + FormatFloat("0.0000", z);  }
```

В даній програмі для виведення значень на екран використовується функція FormatFloat, котра перетворює число в рядок у заданому форматі. Перший параметр цієї функції задає формат, другий – задає числову змінну для перетворення.

Вкладені цикли

Цикли можуть бути вкладені один в другий. При використанні вкладених циклів необхідно складати програму таким чином, щоб внутрішній цикл повністю вкладався в тіло зовнішнього циклу, тобто цикли не повинні перетинатися. В свою чергу, внутрішній цикл може містити свої вкладені цикли. Імена параметрів зовнішнього і внутрішнього циклів повинні бути різними.

Допускаються такі конструкції:

```
for (k=1; k<=10; k++)
```

```
{  for (i=1; i<=10; i++)
   {  for (m=1; m<=10; m++)
      {    ...
       }
    }
 }
```

Задача 4. Обчислити значення $S = \sum_{n=-2}^{m} \frac{n+1}{n} \prod_{k=1}^{n+3} \frac{k}{k+1}$, значення m ввести з екрану.

В цій задачі цикли вкладені один в один, оскільки параметр внутрішнього циклу k залежить від параметра зовнішнього циклу n (k змінюється від 1 до $n + 3$).

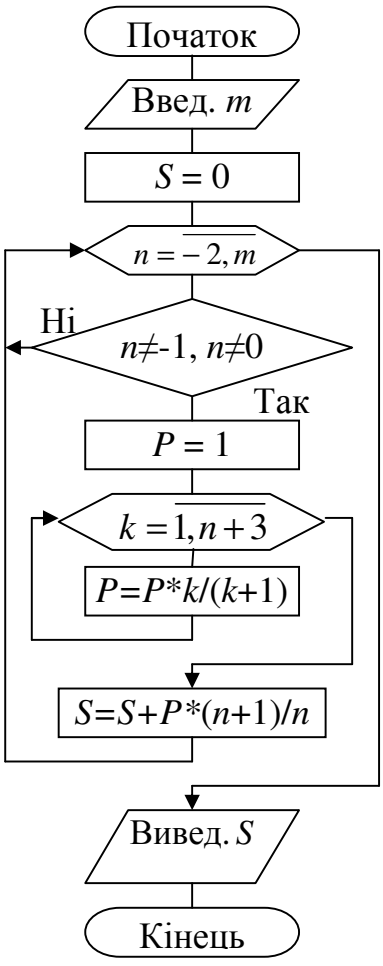


Рисунок 10 – Блок схема до задачі 4

Добуток $\prod_{k=1}^{n+3} \frac{k}{k+1}$ є співмножником доданку і обчислюється у внутрішньому циклі в змінній *P*. Оскільки внутрішній цикл складається лише з одного оператора, то операторні дужки { } необов'язкові.

Перед зовнішнім циклом для обчислення суми слід обнулити змінну S, в якій будуть накопичуватись доданки, а перед зовнішнім циклом для обчислення добутку змінній p присвоїти значення 1.

Оскільки при обчисленні добутку беруть участь тільки цілі числа, то необхідно застосовувати операцію приведення типів (float) k / (k + 1).

Схема алгоритму програми і приклад форми з результатами роботи наведені на рис. 10 і 11.

Текст програми:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{ int n, k, m = StrToInt(Edit1->Text);
  float S = 0, p;
  for (n = -2; n <= m; n++)
     if (n != -1 && n != 0)
      { p = 1;
        for (k = 1; k <= n + 3; k++)
           if  (k != -1 && k != 0)
    p *= (float) k / (k + 1);
    S += (n + 1) * p / n;  }
   Edit2->Text = FloatToStr(S);
  }
```

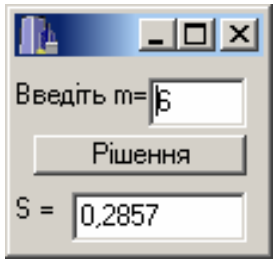


Рисунок 11 – Форма з результатами роботи

Задача 5. Обчислити суму ряду $S = \sum_{i=1}^{7} \frac{2 \cdot x^{2i-1}}{3 \cdot (2i-1)!}$, де $i = 1, 2, \ldots, 7$.

При обчисленні суми *S* необхідно підсумувати сім доданків, для обчислення кожного з яких необхідно організувати вкладений цикл підрахунку факторіалів $(2i-1)!$. В даній програмі кожний доданок обчислюється в окремій змінній *a*.

Оскільки при обчисленні використовується математична функція піднесення до ступеня pow, то на початку у відповідному розділі проекту необхідно підключити математичну бібліотеку math.h. Схема алгоритму програми та приклад форми проекту із результатами роботи наведені на рис. 12 та 13.

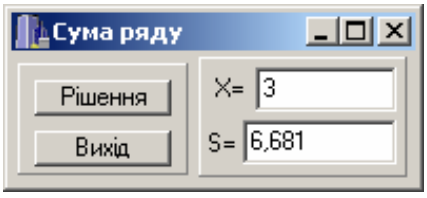


Рисунок 12 – Форма з результатами роботи

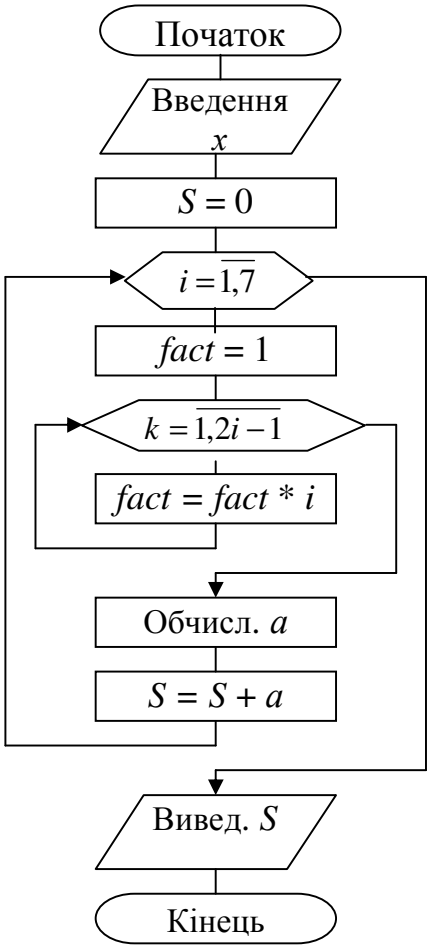


Рисунок 13 – Блок-схема до задачі 5

Текст програми:

. . . . .

```
#include <math.h>
void __fastcall TForm1::Button1Click(TObject
Sender)
{ int i, k, fact;
  float S = 0, a, x = StrToFloat (Edit1->Text);
  for (i = 1; i <= 7; i++)
      { fact = 1;
       for (k = 1; k <= 2 * i - 1; k++)  fact = fact * k;
         a = 2 * pow(x, 2 * i - 1) / (3 * fact);
         s += a;
       }
  Edit2->Text = FloatToStrF(s, ffGeneral, 4, 3);  }
```

Табулювання функцій. Побудова графіків

Досить часто при дослідженні функцій виникає необхідність побудови графіка $y = f(x)$, для чого слід побудувати таблицю значень $y = f(x)$ на заданому проміжку $x \in [A, B]$ із кроком h. Складемо програму табулювання функції $y = f(x)$ при зміненні x від A до B із кроком h, коли $f(x) = \sqrt[3]{\cos^2|x|}$.

Для побудови графіка функції розташуємо на формі компонент Chart, розміщений на закладці Additional. Для налагодження компонента Chart слід двічі клацнути на зображенні цього компонента. В діалоговому вікні, яке з'явиться, слід клацнути по кнопці Add. Із додаткового діалогового вікна виберемо тип графіка Line. Для збільшення розміру графіка на формі, можна видалити відображення «легенди», для чого треба перейти на закладку Legend, в позиції Visible видалити «галочку». З аналогічною метою можна перейти на закладку Titles і видалити надпис TChart, а можна просто змінити надпис графіка, наприклад на «Графік функції».

Приклад вигляду форми табулювання функції може мати вид, показаний на рис. 14.

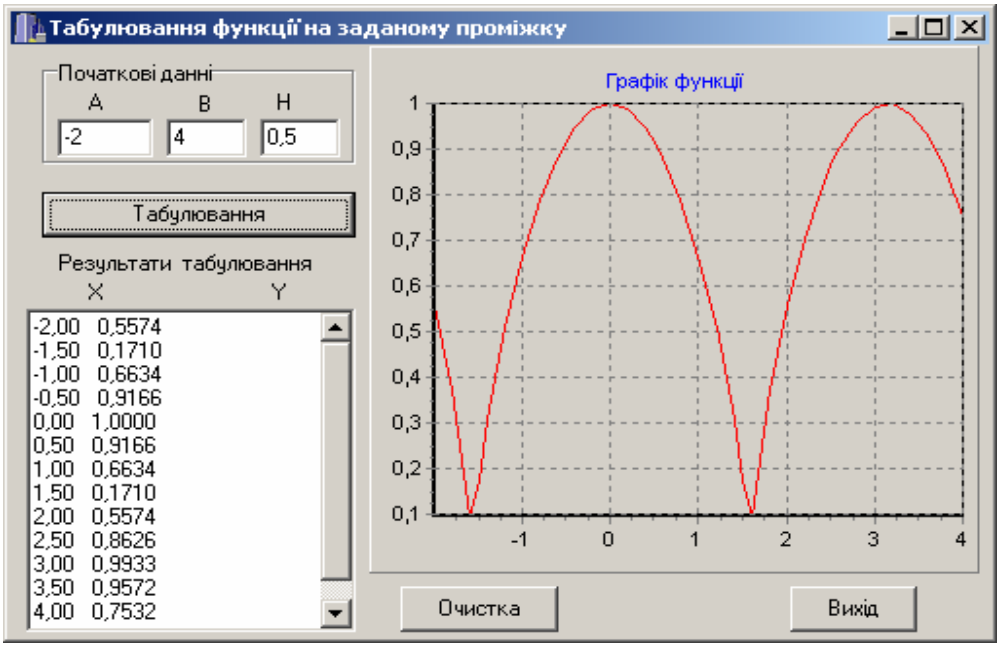


Рисунок 14 – Форма проекту "Табулювання функції"

В наведеному нижче прикладі програми табулювання заданої функції початковими даними, які слід ввести із компонентів Edit1, Edit2 і Edit3, є початкове A і кінцеве B значення x, а також крок зміни $x - h$.

В циклі, починаючи з x=A і допоки буде істинна умова x<=B+0.1*h, обчислюється значення функції y та виводиться її числове значення в компонент Memo1 та на графік в компонент Series1, після чого x збільшується на значення кроку x+=h. Невелика величина 0.1*h, яка не перевищує величини кроку, в умові вико-

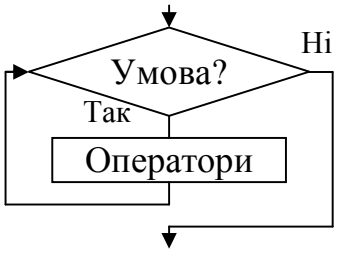
ристовується з тієї причини, що при обчисленні існує так зване накопичення похибки обчислень, наприклад, коли 4 ≅ 3,9999(9). В такому випадку останнє значення *x* і y не виводилося би на форму. Виведення точок на графік здійснюється за допомогою методу AddXY(x, y, "", clRed). Чотири параметри в дужках для цього методу задають параметри точки, що виводиться: перше значення x – координату по горизонтальній осі, друге значення y – координату по вертикальній осі, третій параметр задає параметри відображення числових значень на осях (двоє пустих лапок "" означають, що підписи будуть формуватися автоматично), четвертий параметр задає колір точки. Колір (color) указується після символів cl з великої літери, так clRed задає червоний колір, clBlue – синій, clGreen – зелений тощо.

Текст програми:

```
#include <math.h>
. . . . .
void __fastcall TForm1::Button1Click(TObject *Sender)
{ float A, B, h, x, y;
  A = StrToFloat (Edit1->Text);
  B = StrToFloat (Edit2->Text);
  h = StrToFloat (Edit3->Text);
  for (x = A; x <= B + 0.1 * h; x += h)
   { y = pow(pow(cos(fabs(x)), 2), 1./ 3);
     Memo1->Lines->Add(FormatFloat("0.00", x)+"  "+FormatFloat("0.0000",y));
     Series1 -> AddXY(x, y, “”, clRed);
}   }
```

Оператори циклу з передумовою while та післяумовою do-while

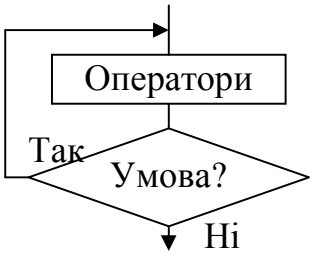
while (*умова*) { *послідовність операторів;*};



Послідовність операторів циклу виконується допоки *умова* істинна (true, має ненульове значення), а вихід із циклу здійснюється, коли *умова* стане неправдивою (false, матиме нульове значення).

Спочатку обчислюється і перевіряється *умова*. Якщо вона не виконується (false), то *послідовність операторів* не виконується і керування передається на наступний оператор програми. Якщо *умова* істинна (не нуль), то виконується *послідовність операторів*. Повторення виконання тіла оператора відбувається допоки *умова* залишається істиною.

do{ *послідовність операторів;*} **while** (*умова*);



Послідовність операторів виконується один або декілька раз допоки *умова* стане неправдивою (false, дорівнює нулю). Оператор циклу do-while використовується в тих випадках, коли необхідно виконати тіло циклу хоча б один раз, оскільки перевірка умови здійснюється після виконання операторів.

Спочатку виконується *послідовність операторів*, тоді обчислюється і перевіряється *умова*. Якщо *умова* не виконується, то оператор завершує виконання і передає керування наступному оператору в програмі. Якщо *умова* істинна (ненульова), то тіло оператора виконується знову і знову перевіряється *умова*. Виконання *послідовності операторів* продовжується допоки *умова* не стане неправдивою.

Якщо тіло циклу складається із одного оператора, то операторні дужки {} необов'язкові. Оператори while і do-while можуть також завершитись при виконанні операторів break, goto, return всередині тіла циклу.

Оператор циклу виду for (*вираз1*; *вираз2*; *вираз3*) *оператор*;
може бути замінено оператором while наступним чином:

вираз1;
while (*вираз2*)
{ *оператори*; *вираз3*; }

Так само, як і при виконанні оператора for, в операторі while спочатку відбувається перевірка умови. Через це оператор while зручно використовувати в ситуаціях, коли тіло оператора не завжди треба виконувати.

Так, цикл for із попереднього прикладу табулювання з використанням оператора циклу while може виглядати наступним чином:

```
. . . . .
x = A;
while(x <= B + 0.1 * h)
 { y = pow(pow(cos(fabs(x)), 2), 1./ 3);
   Memo1->Lines->Add(FormatFloat("0.00",x)+ "   " + FormatFloat("0.0000",y));
   Series1->AddXY(x, y, "", clRed);    x += h; }
. . . . .
```

Той же цикл з використанням оператора do-while буде мати вигляд:

```
x = A;
do { y = pow(pow(cos(fabs(x)), 2), 1./ 3);
  Memo1->Lines->Add(FormatFloat("0.00", x) + "  " + FormatFloat("0.0000",y));
  Series1->AddXY(x, y, "", clRed);   x += h ;  }
while(x <= B + 0.1 * h) ; …
```

В обох випадках початкове значення x=A має бути розміщено до циклу, а в циклі необхідно передбачити оператор x+=h для змінення значення x, інакше умова виходу із циклу ніколи не буде виконана і відбувається зациклення. Всередині операторів циклу можна використовувати локальні змінні, які повинні бути оголошені з визначенням відповідних типів. Розглянемо роботу різних операторів циклу на прикладі обчислення суми всіх непарних чисел у діапазоні від 10 до 100.

1) з використанням оператора for

```
int s=0;  for (int i=11; i<100; i+=2) s+=i;
чи  int i,s;   for (i=11, s=0; i<100;  s+=i++, i++);
чи  int i,s;   for (i=11, s=0; i<100; s+=i, i+=2);
```

2) із використанням оператора while

int s=0, i=11; while (i<100) {s+=i; i+=2;}

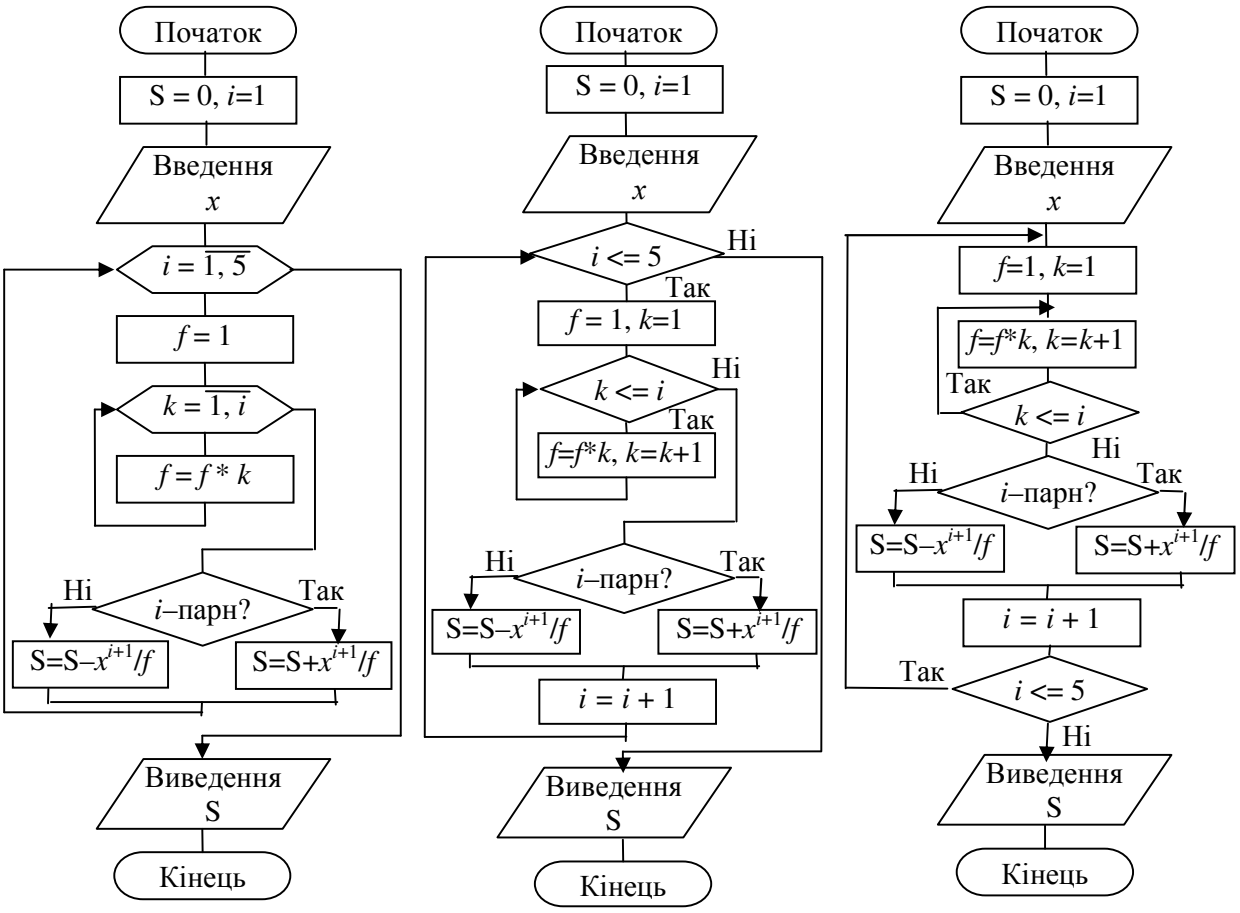
3) із використанням оператора do-while

int s=0, i=11; do {s+=i; i+=2;} while (i<100);

Задача 6. Обчислити суму знакозмінного ряду $S = \sum_{i=1}^{5} \frac{(-1)^i x^{i+1}}{i!}$ трьома варіантами, використовуючи різні оператори циклу.

Тут $(-1)^{i+1}$ при непарних значеннях i (1, 3, 5, …) дорівнює 1, а при парних значеннях i (2, 4, 6, …) – –1, тобто розглядається знакозмінний ряд. Через це у програмі слід передбачити перевірку на парність (i%2==0).

Схеми алгоритмів і приклад форми з результатами роботи програми наведені на рис. 15 і 16.



а б в

Рисунок 15 – Блок-схеми для різних операторів циклу:
а – for; б – while; в – do-while

Приклади програм із використанням різних операторів циклу:

---

void __fastcall TForm1::Button1Click(TObject *Sender) // for

```
{  int i, f, k;   float S = 0, x = StrToFloat(Edit1->Text);
   for (i = 1; i <= 5; i++)
```

```
    {  for (k = 1,f = 1; k <= i; k++) f *= k;
       if (i % 2 == 0) S += pow(x, i+1) / f;
         else S -= pow(x, i+1) / f ;
    }   Edit2->Text = FloatToStr(S); }
void __fastcall TForm1::Button2Click(TObject *Sender) // while
  { int i=1, f, k;
    float S = 0, x = StrToFloat(Edit1->Text);
    while(i <= 5)
     {  f = 1; k = 1;
       while(k <= i) { f *= k; k++; }
       if(i % 2 == 0) S += pow(x, i + 1) / f;
          else S -= pow(x, i + 1) / f ;
       i++;
     }  Edit3->Text = FloatToStr(S); }
```

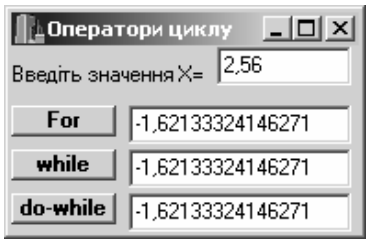


Рисунок 16 – Форма з результатами роботи

```
void __fastcall TForm1::Button3Click(TObject *Sender) // do-while
  { int i = 1, f, k;
    float S = 0, x = StrToFloat(Edit1 -> Text);
    do { f = 1; k = 1;
         do { f *= k; k++; } while(k <= i);
         if (i % 2 == 0) S += pow(x, i+1) / f;
            else  S -= pow(x, i+1) / f ;
         i++;
     } while( i <= 5);
     Edit4 -> Text = FloatToStr(S); }
```

Задача 7. Обчислити суму ряду $S = \sum_{k=1}^{\infty} \frac{x^{2k+1}}{(2k-1)!}$, підсумовуючи члени ряду, значення яких за модулем перевищують задану точність $\varepsilon = 10^{-4}$. Визначити кількість доданків. Значення x ($-2 < x < 2$) вводити з клавіатури.

При розробці програми для розв'язання цієї задачі недоцільно використовувати оператор циклу з параметром for, оскільки наперед невідома кількість повторень такого циклу. Доцільним буде використання оператора циклу з післяумовою do-while, оскільки на момент першої перевірки умови вже необхідно знати значення першого доданку.

Схеми алгоритмів і приклад форми з результатами роботи програми наведені на рис. 17 і 18.

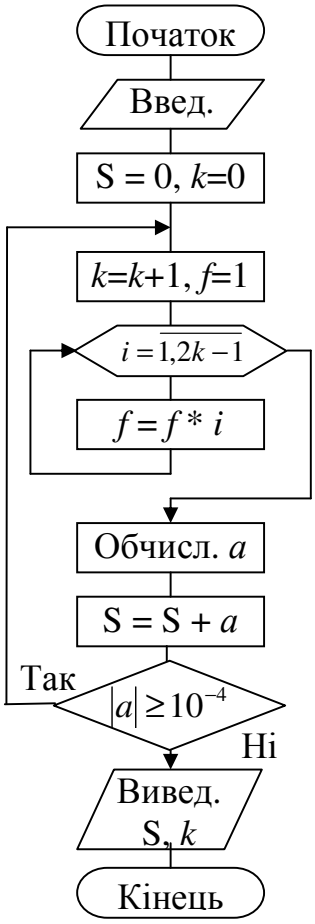


Рисунок 17 – Блок-схема до задачі 7

Текст програми:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
    { float x, a, s=0;    int f, i, k=0;
```

```
 x=StrToFloat(Edit1->Text);
 do  { k++;
       for(i=1,f=1; i<=2*k-1; i++)f*=i;
       a=pow(x, 2*k+1)/f;
       s+=a;     }
while(fabs(a)>=1e-4);

Edit2->Text=FloatToStr(s) ;
Edit3->Text=IntToStr(k); }
```

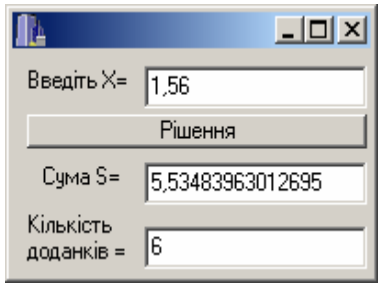


Рисунок 18 – Форма з результатами роботи

Слід зауважити, що організовувати нескінченні цикли можна і з використанням оператора for з пустим умовним виразом, тоді для виходу із циклу зазвичай використовують додаткову умову і оператор break.

```
Приклад        for ( ; ; )
                  { ...
                    ... break;
                    ...          }
```

Задача 8 Обчислити суму ряду $y = \sum_{k=1}^{\infty} \frac{(-1)^k x^{2k}}{2k(2k-1)!}$, підсумовуючи члени ряду, значення яких за модулем більше заданої точності ε. Визначити кількість доданків. Значення x $(-2 < x < 2)$ і $\varepsilon = 10^{-4}$ ввести з клавіатури.

Наведемо два способи розв'язання цієї задачі. Для наочності і контролю правильності окремо виведемо усі доданки. Множник $(-1)^k$ для непарних k = 1, 3, 5, …, дорівнює (–1), а для парних k = 2, 4, …, – дорівнює 1. Таким чином представлений ряд є знакозмінним, де усі непарні доданки будуть від'ємними, а усі парні – зі знаком «+». У програмі оператор if(k%2==1) u = -u; для непарних k змінює знак доданка.

Слід зауважити, що множник $(-1)^{k-1}$ чи $(-1)^{k+1}$ є тільки в знакозмінних рядах із додатними непарними і від'ємними парними доданками.

Приклад форми з результатами роботи програми наведено на рис. 19.

Перший спосіб розв'язання:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
 { float x, y = 0, u, eps;    Memo1->Clear();
  x = StrToFloat(Edit1->Text);
  eps = StrToFloat(Edit2->Text);
  int i, f, k = 1;
  do
```

```
  { for(i = 1, f = 1; i <= 2 * k - 1; i++) f *= i;
   u = pow(x, 2 * k) / (2 * k * f);
   if (k % 2 == 1) u = -u;
   Memo1->Lines->Add(IntToStr(k) + ") " +
FormatFloat("0.00000", u));
   y += u;
   k++;
  } while (fabs(u) >= eps) ;
 Edit3->Text = FloatToStr(y);
 Edit4->Text = IntToStr(k - 1);
}
```

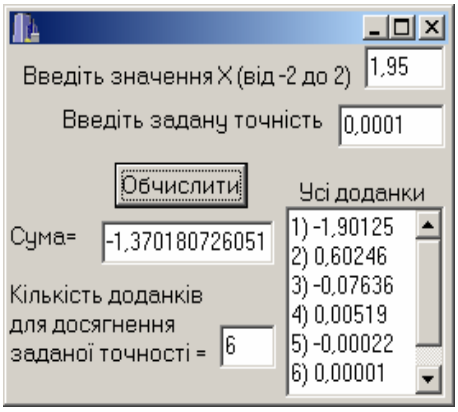


Рисунок 19 – Форма з результатами роботи

Щоб зробити алгоритм програми більш оптимальним, можна його удосконалити, але для цього треба обчислити рекурентний множник. Це дозволить в даній програмі позбавитись вкладеного циклу для обчислення факторіалу і оператора перевірки на непарність k. Зупинимось більш докладно на виведенні рекурентної формули.

Представимо суму ряду $y = \sum_{k=1}^{\infty} \frac{(-1)^k x^{2k}}{2k(2k-1)!}$ у вигляді $y = \sum_{k=1}^{\infty} u_k$, де $u_k = \frac{(-1)^k x^{2k}}{2k(2k-1)!}$.

Рекурентний множник – це співвідношення двох поряд розташованих членів ряду:

$$u_2 = R \cdot u_1;$$
$$u_3 = R \cdot u_2;$$
$$\dots$$
$$u_k = R \cdot u_{k-1},$$

звідки $\boxed{R = \frac{u_k}{u_{k-1}}}$. (1)

Для визначення R в формулу (1) слід підставити $u_k = \frac{(-1)^k x^{2k}}{2k(2k-1)!}$ і u_{k-1}. Для обчислення u_{k-1} підставимо у вираз для u_k замість $k-(k-1)$:

$$u_{k-1} = \frac{(-1)^{k-1} x^{2(k-1)}}{2(k-1)(2(k-1)-1)!} = \frac{(-1)^{k-1} x^{2(k-1)}}{2(k-1)(2k-3)!};$$

$$R = \frac{u_k}{u_{k-1}} = \frac{(-1)^k x^{2k}}{2k(2k-1)!} \cdot \frac{2(k-1)(2k-3)!}{(-1)^{k-1} x^{2(k-1)}} = \frac{(-1)^k \cdot x^{2k} \cdot (2k-2) \cdot (2k-3)!}{(-1)^{k-1} \cdot x^{2k-2} \cdot 2k \cdot (2k-1)!} =$$

$$= \frac{(-1)^{k-k+1} \cdot x^{2k-(2k-2)} \cancel{(2k-2)}}{2k} \cdot \frac{\cancel{1 \cdot 2 \cdot \ldots \cdot (2k-3)}}{\cancel{1 \cdot 2 \cdot \ldots \cdot (2k-3)} \cdot \cancel{(2k-2)} \cdot (2k-1)} = -\frac{x^2}{2k(2k-1)}.$$

Крім рекурентного множника *R* слід обчислити перший член ряду при $k = 1$:

$$u_1 = \frac{(-1)^1 x^2}{2(2-1)!} = -\frac{x^2}{2}.$$

Другий спосіб розв'язання з використанням рекурентних множників:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{  float x, y, u, r, eps;
   Memo1->Clear();
  x = StrToFloat(Edit1->Text);
  eps = StrToFloat(Edit2->Text);
  int i, f, k = 1;
  u = -x * x / 2;  y = u;
  Memo1->Lines->Add(IntToStr(k) + ") " + FormatFloat("0.00000", u));
  do { k++;
       r = -x*x/(2*k*(2*k-1));
       u *= r ;
       Memo1->Lines->Add(IntToStr(k) + ") " + FormatFloat("0.00000", u));
       y += u;
     }  while(fabs(u) >= eps)  ;
  Edit3->Text = FloatToStr(y);
  Edit4->Text = IntToStr(k);
}
```

Одновимірні масиви

Поняття масиву

М а с и в (array) в програмуванні – це упорядкована сукупність однотипних елементів. Масиви широко застосовуються для зберігання і опрацювання однорідної інформації, наприклад, таблиць, векторів, матриць, коефіцієнтів рівнянь.

Кожен елемент масиву однозначно можна визначити ім'ям та індексами. Ім'я масиву (ідентифікатор) підбирають за тими ж правилами, що й для змінних. Індекси визначають місцезнаходження елемента в масиві. Наприклад, елементи вектора мають один індекс – номер по порядку; елементи матриць чи таблиць мають по два індекси: перший означає номер рядка, другий – номер стовпчика. Кількість індексів визначає вимірність масиву. Наприклад, вектори в програмах – це одновимірні масиви, матриці – двовимірні. В цій лекції розглянемо лише одновимірні масиви.

Оголошення одновимірних масивів

Одновимірний масив оголошується в програмі наступним чином:

тип_даних ім'я_масиву [*розмір_масиву*];

Ім'я_масиву – це ідентифікатор масиву. *Тип_даних* задає тип елементів масиву. Елементами масиву не можуть бути функції і елементи типу void. *Розмір_масиву* в квадратних дужках задає кількість елементів масиву. На відміну від мови Pascal в С не перевіряється вихід за межі масиву, тому, щоб уникнути помилок у програмі, слідкуйте за розмірами оголошених масивів. Значення *розмір_масиву* при оголошенні масиву може бути не вказано в наступних випадках:

- при оголошенні масив ініціюється,
- масив оголошено як формальний параметр функції,
- масив оголошено як посилання на масив, явно визначений в іншому файлі.

Використовуючи ім'я масиву та індекс, можна звертатися до елементів масиву:

ім'я_масиву [*значення_ індексу*]

Значення індексів повинні бути в діапазоні від нуля до величини, на одиницю меншу, ніж розмір масиву, указаний при його оголошенні, оскільки в С++ нумерація індексів завжди розпочинається з нуля.

Наприклад:

```
int A[10];
```

оголошує масив з ім'ям А, який містить 10 цілих чисел. А[0] – значення першого елемента, А[1] – другого, А[9] – останнього.

При оголошенні під масив виділяється пам'ять, необхідна для розташування усіх його елементів. Елементи масиву з першого до останнього запам'ятовуються в послідовно зростаючих адресах пам'яті. Між елементами масиву в пам'яті проміжків немає. Елементи масиву запам'ятовуються один за одним поелементно. Так, для розміщення елементів одновимірного масиву int B[5] виділяється по 4 байта під кожен із 5 елементів масиву – тобто 20 байт.

Елементи	B[0]				B[1]				B[2]				B[3]				B[4]			
Байти	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Таким чином, для отримання доступу до і-го елемента масиву В, можна написати B[i]. При цьому параметр і перемножується на розмір типу int та являє собою адресу і-го елемента масиву В від його початку, після чого здійснюється вибірка елемента масиву В за сформованою адресою.

Змінна (ідентифікатор), оголошена як масив, – по суті це покажчик, значення якого є адресою самого першого елемента масиву. Слід зауважити, що адреса ма-

сиву не може бути змінена під час виконання програми, хоча значення елементів може змінюватись.

Наведемо декілька прикладів оголошення масивів:

```
char  N [ 20 ];
int grades [ 125 ];
float mass [ 30 ];
double vector[ 1500 ];
```

Перший з масивів N містить 20 символів. Звертанням до елементів масиву може бути N[0], N[1], ..., N[19].

Другий масив grades містить 125 цілих чисел. Звертанням до елементів такого масиву може бути: grades [0], grades [1], ..., grades[124].

Третій масив mass містить 30 дійсних чисел. Звертанням до елементів масиву може бути mass[0], mass[1], ..., mass[29].

Четвертий масив vector містить 1500 дійсних чисел з подвійною точністю. Звертанням до елементів такого масиву може бути: vector[0], vector[1], ..., vector[1499].

При оголошенні масивів можна елементам масиву (необов'язково всім) присвоювати початкові значення, які в подальшому в програмі можуть бути змінені. Якщо реальна кількість значень, що ініціюються, менше, ніж розмірність масиву, то решта елементів масиву набуває значення 0.

Наприклад,

```
int  a[5] = {9, 33, -23, 8, 1};     // a[0]=9, a[1]=33, a[2]=-23, a[3]=8, a[4]=1
float Q[10]={1.5, -3.8, 10};      //Q[0]=1.5,Q[1]=-3.8,Q[2]=10,Q[3]=Q[4]=…=Q[9]=0
char S[3]="Фото";                 // S[0]="Ф", S[1]="о", S[2]="т", S[3]="о"
char code[ ] = "abc";             // code[0]="a",code[1]="b",code[2]="c", code[3]="\0"
```

В останньому прикладі ініціюється code як масив символів із чотирьох елементів. Четвертим елементом є символ "\0", який завершує усі рядки символів. Якщо рядок коротше, ніж оголошений розмір масиву символів, то решта елементів масиву ініціюється нулем (символом "\0").

Масиви в програмах можна об′являти також із створенням типу користувача:

typedef *тип_даних ім'я_типу* [*розмір_масиву*];
ім'я_типу ім'я_масиву ;

Наприклад, створимо тип з ім'ям mass як масив із 10-ти додатних цілих чисел та оголосимо два масиви – a і b – створеного типу:

```
typedef unsigned short mass[10];
mass a,b;
```

Для оголошення масивів можна використовувати типізовані константи-масиви, які дозволяють водночас і оголосити масив, і задати його значення як константи:

const *тип_даних ім'я_масиву* [*розмір_масиву*] =
{*значення_елементів_масиву*};

Слід пам'ятати, що змінювати значення елементів констант-масивів неможна. Приклади:

```
const  int  arr[5] = {9, 3, -7, 0, 123};          // масив із 5 цілих чисел
const  float  f [5] = {1.5, 6, 8, 7.4, -1.125};  // масив із 5 дійсних чисел
const    C[5] = {15, -6, 546};             // масив із 5 цілих чисел типу int,
                                  // де C[0]=15, C[1]=-6, C[2]=546, C[3]=0 и C[4]=0
```

Виведення елементів одновимірного масиву

Виводити значення масивів можна у файл чи на форму, використовуючи різноманітні компоненти C++ Builder. При цьому виводити значення елементів масивів можна лише поелементно, для чого слід організувати цикли зі зміною значення індексу. Тут розглянемо організацію виведення одновимірних масивів на форму за допомогою компонентів Edit, Label, Memo, ListBox та функції ShowMessage.

В наступних прикладах будуть використані такі змінні:

```
float A[10];
int i; AnsiString st;
```

Виведення в Edit

В компонент Edit можна виводити одновимірні масиви, відокремлюючи елементи прогалинами (" ") чи іншими символами. Кількість елементів масиву, котрі можна побачити, обмежена довжиною компонента Edit на формі.

Приклад фрагмента програми виведення масиву A:

```
st = "" ;                  // очищення рядка st;
for(i=0; i<=9; i++)    // початок циклу за індексами масиву для накопичення
  {  st += FormatFloat("0.00", A[i])+"  ";  }  // в рядку значень масиву;
Edit1->Text=st;     // виведення в компонент Edit1 сформованого рядка
```

Виведення у Label

В компонент Label можна виводити масиви, відокремлюючи елементи прогалинами (" ") чи символами переходу до нового рядка ("\13"). Виведення одновимірного масиву в рядок зорганізовують за тими ж правилами, що і в компонент Edit, лише в програмі замість Edit1->Text слід записати Label1->Caption (наприклад, Label1->Caption=st;). Для виведення одновимірного масиву у стовпчик слід властивість WordWrap компонента Label встановити в True замість False, яке стоїть за замовчуванням.

Виведення у вікно повідомлень

Виведення у вікно повідомлень за допомогою функції ShowMessage зорганізовують так само, як і в попередніх прикладах, лише замість оператора присвоєння слід записати оператор виклику функції. Наприклад, замість оператора Edit1->Text = st;
слід записати: ShowMessage(st);

Виведення у Memo

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

Приклад фрагмента програми виведення масиву A (у стовпчик):

```
Memo1.Clear();          // очищення компонента
for(i=0; i<=9; i++)     // початок циклу за індексами масиву
Memo1->Lines->Add(FormatFloat("0.00",A[i])); //виведення елемента масиву
```

Виведення у ListBox

Виведення масивів за допомогою компонента ListBox зорганізовують так само, як і для компонента Memo, лише замість Memo слід записати компонент ListBox.

Наприклад, замість оператора

```
Memo1->Lines->Add(FormatFloat("0.00",A[i]));
```

слід записати

```
ListBox1->Items->Add(FormatFloat("0.00",A[i]));
```

Введення елементів одновимірного масиву

Як і при виведенні масивів, при їх введенні слід організовувати цикли змінювання зі зміною значення індексу. Розглянемо введення елементів масивів використовуючи компоненти Memo і ListBox.

Введення з вікна Memo

За допомогою компонента Memo можна вводити масиви, при виконанні програми, так і при конструюванні форми проекту програми через вікно властивості Lines (для переходу до нового рядка при введенні значень треба натиснути клавішу <Enter>).

Приклад фрагмента програми введення значень елементів одновимірного масиву A (в кожному рядку по одному числу):

```
for(i=0; i<=9; i++)  A[i]=StrToFloat(Memo1->Lines->Strings[i]);
```

Введення з ListBox

За допомогою компонента ListBox можна вводити масиви так само, як і з компонентом Memo, лише замість властивості Lines використовувати властивість Items.

Приклади програм опрацювання одновимірних масивів

Задача 9. Розробити схему алгоритму та проект програми для введення елементів одновимірного масиву з 8 дійсних елементів та обчислення суми усіх елементів масиву.

Схему алгоритму програми і приклад форми із результатами роботи наведено на рисунках 20 і 21. На формі розташовані компоненти Memo1, Edit1, Label1, Button1, Button2, Button3, а їх нові властивості наведені в табл. 4.

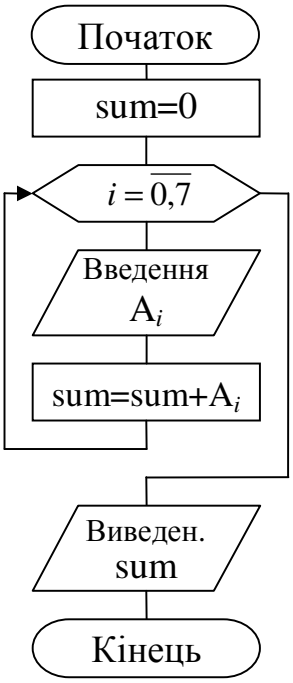


Рисунок 20 – Блок-схема до задачі 9

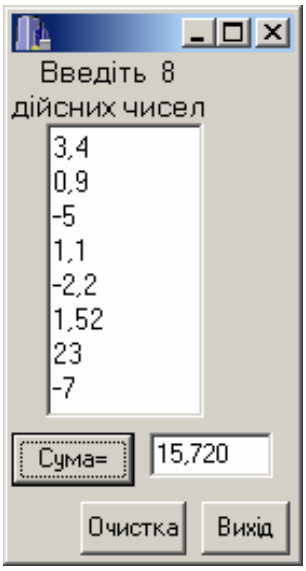


Рисунок 21 – Форма із результатами роботи

Таблиця 4 – Нові властивості компонет

Компоненти	**Властивості**	**Нові значення**
Label1	Caption	Введіть 8 дійсних чисел
Button1	Caption	Сума =
Button2	Caption	Очистка
Button3	Caption	Вихід

Текст програми:

```
void __fastcall TForm1::Button1Click(TObject *Sender) // Сума
{ float A[8], sum = 0;
  for(int i = 0; i < 8; i++)
   {  A[i] = StrToFloat(Memo1->Lines->Strings[i]);
     sum += A[i];
    }
  Edit1->Text = FormatFloat("0.000", sum)      ; }
```

```
void __fastcall TForm1::Button3Click(TObject *Sender) // Очистка
{  Memo1->Clear();
   Edit1->Clear();    }
```

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

Задача 10. Скласти схему алгоритму і проект програми для обчислення елементів одновимірного масиву з 15 елементів за формулою $A_i = \lg(i) + \mathrm{tg}(2^i)$, де i=1, 2, .., 15, та їх виведення на форму, а також визначення мінімального елемента та його порядкового номера.

Схеми алгоритмів програми та приклад виду форми із результатами роботи наведено на рис. 22 … 24. На формі розташовані компоненти Memo1, Edit1, Button1, Button2, а їх нові властивості наведено в табл. 5.

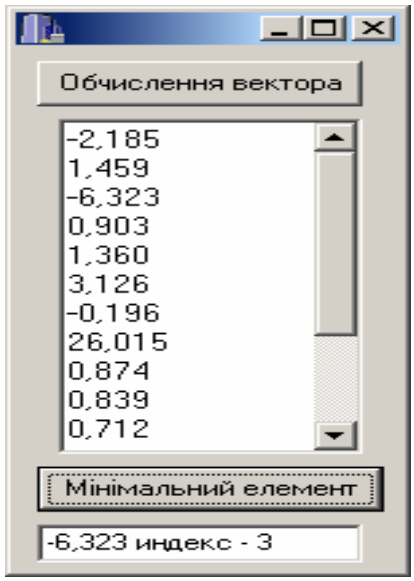


Рисунок 22 – Форма з результатами роботи

Т а б л и ц я 5 – Нові властивості компонентів

Компоненти	**Властивості**	**Нові значення**
Memo1	ScrollBars	ssVertical
Button1	Caption	Обчислення вектора
Button2	Caption	Мінімальний елемент

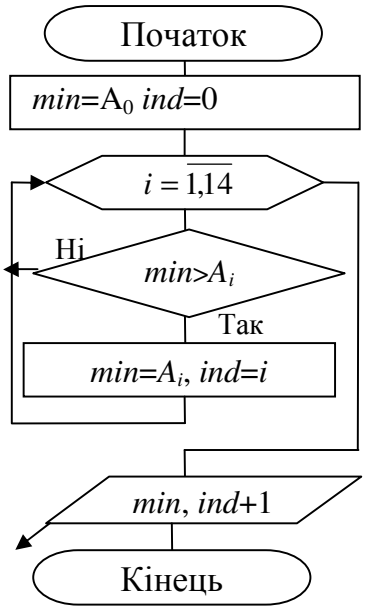


Рисунок 23 – Блок-схема для кнопки «Мінімальний елемент»

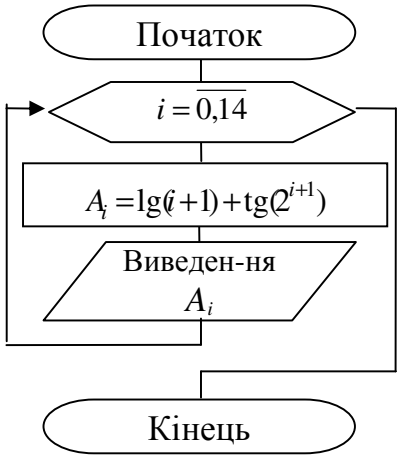


Рисунок 24 – Блок-схема для кнопки «Обчислення вектора»

Текст програми:

```
#include <math.h>
. . . . . .
float A[15] ;  int i;
```

void__fastcall TForm1:: Button1Click (TObject *Sender)

```
{ Memo1->Clear();
  for(i = 0; i < 15; i++)
  {  A[i] = log10(i + 1) + tan(pow(2., i+1));
     Memo1->Lines->Add(FormatFloat("0.000", A[i]));   }  }
void __fastcall TForm1::Button2Click(TObject *Sender)
{ float min = A[0];   int ind = 0;
  for(i = 1; i < 15; i++)
   if(min > A[i])  {   min=A[i]; ind=i;   }
  Edit1->Text = FormatFloat("0.000", min) + " индекс - " + IntToStr(ind + 1);
 }
```

В цій програмі змінні A[15] та і оголошені глобально перед процедурами-обробниками події Click. Це зроблено для того, щоб значення елементів масиву, набуті в одній процедурі, були доступні для опрацювання і в другій процедурі.

Задача 10. Розробити проект програми для обчислення:

- добутку ненульових елементів;
- мінімального елемента і поміняти його місцями з першим елементом вектора;
- кількості від'ємних елементів;
- у послідовності цілих чисел непарні елементи (за значенням, а не за індексом) замінити одиницями, а парні елементи – нулями;
- середньо арифметичного усіх елементів одновимірного масиву

Розв'язання поставлених задач наведемо у фрагментах програм.

```
// «Добуток ненульових елементів»
            …
 float P = 1;              // перед добутком змінна  P=1
  for(i = 0; i < kol; i++)
    if(v[i] != 0) P *= v[i]; // в P накопичуємо добуток ненульових v[i]
            …
// Обмін місцями значень першого і мінімального елементів
              …
{ float min= v[0];//Спочатку вважаємо, що перший елемент є мінімальним.
 int ind=0;
   for(i = 1; i < n; i++)  // В циклі, починаючи з другого елемента, перевіряємо,
     if (v[i] < min)        // якщо є ще менший елемент
     { ind = i; min = v[i]; } //запам'ятовуємо індекс цього елемента і його значення
   v[ind] = v[0]; // На місце мінімального записати значення першого елемента
   v[0] = min;     }   // а на місце першого – значення мінімального
            …
// «Кількість від'ємних елементів»
              …
```

```
 int k = 0;            // Спочатку вважаємо, що від'ємних елементів немає
for(i = 0; i < kol; i++)
 if(v[i] < 0) k++;        // Якщо елемент від'ємний, k збільшується на 1
        …
```

// «Заміна елементів»

```
{ int i;
 for (i = 0; i < n; i++)   // n – кількість елементів масиву
  if(V[i] % 2) V[i] = 1;
    else  V[i] = 0;
}
```

// «Обчислення середньоарифметичного усіх елементів масиву »

```
    …
float  sr, s = 0;
  for(int  i = 0; i < n; i++)
    A[i] = StrToFloat(Memo1->Lines->Strings[i]);
 for (int i = 0; i < n; i++) s += A[i];
 sr = s / n;
 Edit1->Text = FormatFloat("0.000", sr);
        …
```

Задача 12. Розробити схему алгоритму і проект програми для введення до 15-ти цілих чисел із компонента Memo і впорядкувати їх за зростанням.

Сортування елементів масиву зорганізуємо в окремій функції. Схему алгоритму наведено на рис. 25.

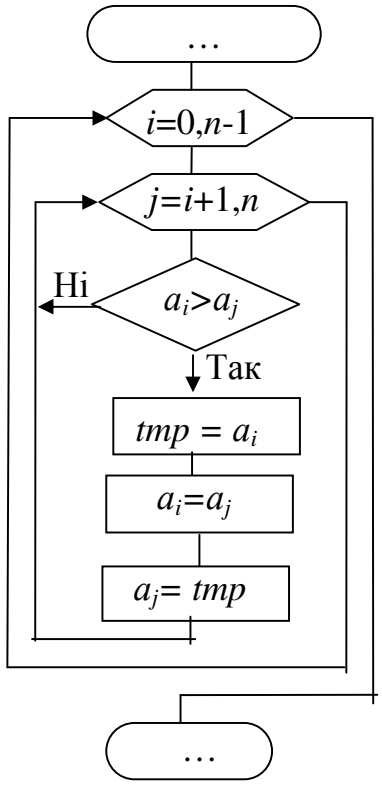


Рисунок 25 – Схема сортування масиву за зростанням

Методів сортування існує багато. В наведеному нижче фрагменті тексту програми запропонований найбільш простий і розповсюджений метод “бульбашки” (bublesort). Алгоритм сортування полягає в обміні місцями порівнюваних елементів масиву. Змінна tmp потрібна для тимчасового зберігання одного з елементів, що переставляються. Якщо в умові порівняння елементів записати знак «>», то елементи будуть сортуватися за зростанням, а при знаку «<» – за спаданням. Потреба в організації двох циклів зумовлена тим, що зовнішній цикл дозволяє перебрати всі елементи масиву для порівняння з іншими елементами, доступ до яких організовано у вкладеному циклі.

Фрагмент текста програми:

```
        …
{  int i, j, tmp;
   for (i = 0; i < n - 1; i++)
```

```
for (j = i + 1; j < n; j++)
  if (a[i] > a[j])
    {  tmp = a[i];
       a[i] = a[j];
       a[j] = tmp;
    }                                    }
   …
```

Контрольні завдання

Метою контрольного завдання є вивчення алгоритмічної мови С++, надбання практичних навичок у складанні алгоритмів рішення задач та реалізації обчислювальних процесів на персональному комп'ютері у середовищі С++Builder*:* Контрольне завдання складається із **6 завдань**. Кожне завдання містить 30 варіантів. Номер індивідуального варіанта відповідає номеру студента у списку групи або визначається викладачем.

Завдання 1

Відповідно до варіанта записати на мові С++ арифметичний вираз, наведений у табл. 6.

Таблиця 6 – Варіанти арифметичних виразів

1	$Z=\frac{2t+y\cos t}{\sqrt{y+4{,}831}}$	**2**	$D=y^2+\frac{0{,}5n+4{,}8}{\sin y}$	**3**	$Q=\frac{\sqrt{k+2{,}6p\sin k}}{x-d^3}$
4	$F=\ln(d)+\frac{3{,}5d^2+1}{\cos 2y}$	**5**	$R=\frac{\sin(2t+1)^2+0{,}3}{\ln(t+y)}$	**6**	$L=\cos^2 c+\frac{3t^2+4}{\sqrt{c+t}}$
7	$U=\frac{\ln(k-y)+y^4}{e^y+2{,}355k^2}$	**8**	$A=\frac{\sin(2y+h)+h^2}{e^h+y}$	**9**	$R=\frac{\sin^2 y+0{,}3d}{e^y+\ln(d)}$
10	$G=\frac{9{,}33w^3+\sqrt{w}}{\ln(y+3{,}5)+\sqrt{y}}$	**11**	$P=\frac{e^{y+2{,}5}+7{,}1h^3}{\ln\sqrt{y+0{,}04h}}$	**12**	$U=\frac{\ln(2k+4{,}3)}{e^{k+y}+\sqrt{y}}$
13	$D=\frac{7{,}8a^2+3{,}52t}{\ln(a+2y)+e^y}$	**14**	$F=\frac{2\sin(0{,}354y+1)}{\ln(y+2j)}$	**15**	$T=\frac{\sin(2u)}{\ln(2y+u)}$
16	$L=\frac{0{,}81\cos i}{\ln(y)+2i^3}$	**17**	$W=\frac{4t^3+\ln(r)}{e^{y+r}+7{,}2\sin r}$	**18**	$G=\frac{e^{2y}+\sin(f)}{\ln(3{,}8y+f)}$
19	$N=\frac{m^2+2{,}8m+0{,}355}{\cos 2y+3{,}6}$	**20**	$H=\frac{y^2-0{,}8y+\sqrt{y}}{23{,}1n^2+\cos n}$	**21**	$Z=\frac{\sin(p+0{,}4)^2}{y^2+7{,}325p}$
22	$T=\frac{2{,}37\sin(t+1)}{\sqrt{4y^2-0{,}1y+5}}$	**23**	$R=\frac{\sqrt{\sin^2 y+6{,}835}}{\ln(y+k)+3y^2}$	**24**	$W=\frac{0{,}004v+e^{2y}}{e^{\frac{y}{2}}}$
25	$V=\frac{(y+2w)^3}{\ln(y+0{,}75)}$	**26**	$E=\frac{\ln(0{,}7y+2q)}{\sqrt{3y^2+0{,}5y+4}}$	**27**	$T=\frac{0{,}355h^2-4{,}355}{e^{y+h}+\sqrt{2{,}7y}}$
28	$S=\frac{4{,}351y^3+2t\ln(t)}{\sqrt{\cos 2y+4{,}351}}$	**29**	$K=\frac{2t^2+3l+7{,}2}{\ln(y)+e^{2l}}$	**30**	$N=\frac{3y^2+\sqrt{y+1}}{\ln(p+y)+e^p}$

Завдання 2

Скласти схему алгоритму й проект роботи програми в середовищі C++ Builder*:* для обчислення функції *y=f(x)* відповідно до варіантів, зазначених у табл. 7.

Примітка. Значення першого з двох параметрів, які наведено в цій таблиці, задати як *константу*, значення другого – *ввести* за допомогою компонента Edit.

Таблиця 7 – Варіанти функцій з лінійною структурою

№ вар.	Функція Y=F(x)	Значення параметрів
1	$y = a\sin^2 b + b\cos^2 a$; $a = \sqrt[3]{\lvert b+c\rvert}$; $b = \sqrt{x}$	x=1.52 ; c=5
2	$y = a^2 + b^2$; $a = \ln\lvert x\rvert$; $b = e^k + a$	x=5.3 ; k=3
3	$y = e^x + 5.8^c$; $c = a^2 + \sqrt{b}$; $a = b^3 + \ln\lvert b\rvert$	x=2.5 ; b=7
4	$y = \sqrt[3]{\lvert a-b\rvert}$; $a = \lg x$; $b = \sqrt{x^2 + t^2}$	x=1.7 ;t=3
5	$y = a^3 / b^2$; $a = e^{\sqrt{\lvert x\rvert}}$; $b = \left(\sin p^2 + x^3\right)$;	x=2.1 ; p=2
6	$y = p^2 + t^4$; $p = x^2 - \sqrt{\lvert x\rvert}$; $t = \sqrt[3]{x + a^2}$	x=4 ; a=3.7
7	$y = c^3 / \cos c$; $c = a^2 + b^2$; $a = \sqrt{\lvert x\rvert} + e^{\sqrt{b}}$	x=–11 ; b=12.5
8	$y = \sin^3(a+b)$; $a = t^3 + \sqrt{b}$; $b = \lg^2\lvert x\rvert$	x=10.9 ; t=2
9	$y = arctg^3 x^2$; $x = p + k$; $k = \sqrt{p + t^2}$	t=4.1 ; p=3
10	$y = \cos^2(a + \sin b)$; $a = \sqrt{\lvert x\rvert}$; $b = x^4 + m^2$	m=2 ; x=1.1
11	$y = \sin^3 a + \cos^2 x$; $a = c + k^2$; $c = arctg\lvert x\rvert$	k=7.2 ; x=5
12	$y = e^{\sqrt{\lvert x\rvert}} + \cos x$; $x = a + c^3$; $a = \sin^5 b$	b=3 ; c=1.7
13	$y = a\cos x - b\sin x$; $x = \sqrt[3]{a-b}$; $a = t^2 b$	t=2.2 ; b=3
14	$y = \sqrt{x}\sin a + \sqrt{b}\cos x$; $a = \lg\lvert x\rvert$; $b = x + p^3$	x=11 ;p=2.6
15	$y = \lg a / \lg b$; $a = \sqrt{x^2 + b^2}$; $x = e^b + N$	N=9.1 ; b=3
16	$y = \ln\lvert x + t\rvert$;$x = t^2 + p$;c	M=3.8 ; p=2
17	$y = e^{a+b}$; $a = \lg\lvert t + b^2\rvert$; $t = b^2 + \sqrt{bx}$	b=3 ; x=5.2
18	$y = \sqrt[3]{x^2 + c^2}$;$x = e^{mk}$;$c = \cos^2 m + k^2$	k=2 ; m=1.8
19	$y = p + v^3$; $p = \lg\lvert x\rvert$; $v = \sqrt{x+t} / \left(t^2 + x^2\right)$	x=5 ; t=1.8
20	$y = x^3 / t^2$; $x = e^{\sqrt{p+a}}$; $t = p^3 + a^3$	a=2 ; p=2.6

№ вар.	Функція Y=F(x)	Значення параметрів
21	$y = c^2 + \sqrt{\lvert a\rvert}\ ;\ c = \lg\lvert b\rvert\ ;\ a = (b+x)^3$	b=7 ; x=2
22	$y = arctg^2\lvert x\rvert; x = t^3 + b^2; t = b^3 + e^{\sqrt{q}}$	q=2 ; b=1.8
23	$y = v^3 + \cos^2 w\ ;\ v = \cos^2 a\ ;\ w = \sqrt{a + \lvert x\rvert}$	x=2.9 ; a=-0.9
24	$y = x^2 + \sqrt[3]{\lvert x\rvert}\ ;\ x = \cos^2 b + \sin^2 a\ ;\ a = \sqrt{b + t^2}$	b=7.1 ; t=2
25	$y = \sin^3 x + \cos x^2\ ;\ x = \lg\lvert ct\rvert\ ;\ c = t^2 + \sqrt{a}$	t=-3 ; a=8.8
26	$y = \lg^2\lvert x + a\rvert\ ;\ x = \sqrt{a+b}\ ;\ a = e^{t+b}$	t=2 ; b=1.8
27	$y = arctg^3\lvert p\rvert\ ;\ p = \sqrt{x^2 + a^2}\ ;\ x = \sqrt{a} + \sqrt{b}$	a=7 ; b=2.3
28	$y = \ln^2(p+t)^2\ ;\ p = e^{\sqrt{t}}\ ;\ t = x^2 + \sqrt{\lvert n\rvert}$	x=3 ; n=-1.9
29	$y = \cos^3 x + \lvert a\rvert; x = e^b; b = \lvert a\rvert + \sqrt{a + p^2}$	a=-4 ; p=3
30	$y = \sin^4\left(a^2 + b^2\right)\ ;\ a = \sqrt{b+t}\ ;\ t = b^2 + k^3$	b=2 ; k=1.8

Завдання 3

Скласти структурну схему алгоритму та програму обчислення значень ***y*** за формулами, зазначеними у таблиці 8 . При складанні програми передбачити: введення значень ***x*** з клавіатури (для перевірки), виведення на форму проекту всіх варіантів значень вхідних даних і результатів обчислення.

Примітка. При виконанні програми на комп'ютері треба перевірити роботу програми для кожної умови розгалуження (для кожної формули значень ***y***).

Таблиця 8 – Варіанти функцій із розгалуженою структурою.

N вар	Формули для *y*	Параметри
1	$y = \begin{cases} 2u + \sqrt[3]{a \cdot c}, & u \le a - c^2 \\ \ln(a \cdot c) + cos^2 u, & u > a - c^2 \end{cases}$	$a = 3.6;$ $c = 2.1;$ $u = \sin x$
2	$y = \begin{cases} abx\cos^2 zx, & x < 3{,}5a \\ (a+bx)^2 - \ln(zx), & 3{,}5a \le x \le b \\ \sqrt{(a+bx-zx^2)}, & x > b \end{cases}$	$a = 0.4;$ $b = 2.3;$ $z = e^{2x}$

N вар	Формули для *y*	Параметри
3	$y=\begin{cases}\sin(bm+\cos nx), & \lvert bm\rvert > n^2\\ \cos(bm-\sin x), & \lvert bm\rvert < n^2\\ \sqrt{e^{\lvert Lnx\rvert}+\sqrt{\lvert bmx\rvert}}, & \lvert bm\rvert = n^2\end{cases}$	$b=-1.6;$ $m=0.9;$ $n=-1.4$
4	$y=\begin{cases}a\sin^2 x+bsoszx, & x<-\ln(a)\\ a^b-sos^3(a+zx), & -\ln(a)<x\le b\\ \sqrt{2.5a^3+(b-zx^2)^6}, & x>b\end{cases}$	$a=0.2;$ $b=0.5;$ $z=e^{ax}$
5	$y=\begin{cases}\sin(e^{a+b})+x^2, & e^{a+b}>e^c\\ arctg(abc)+\sqrt[3]{x}, & e^{a+b}=e^c\\ \cos\left(\sqrt{\lvert x+abc\rvert}\right), & e^{a+b}<e^c\end{cases}$	$a=-4,2;$ $b=5.3;$ $c=1.5$
6	$y=\begin{cases}2.8cin^2ax-bx^3z, & x<a\\ z\cos(ax+b)^2+\ln(z), & a\le x\le b^2\\ e^{2,5ax}+zabx, & x>b^2\end{cases}$	$a=-5;$ $b=2.5;$ $z=\ln\lvert bx^3\rvert$
7	$y=\begin{cases}xe^a+e^{\lvert bc\rvert}, & \lvert 1-b^2\rvert = a+c\\ \sin^2 ax+\cos bc, & \lvert 1-b^2\rvert > a+c\\ \sqrt{ab^4+\sqrt{cx,}} & \lvert 1-b^2\rvert < a+c\end{cases}$	$a=3.2;$ $b=-0.7;$ $c=2.2$
8	$y=\begin{cases}\ln\lvert mx+n\rvert, & k^2>m+n\\ e^{Ln\lvert mx-n\rvert}, & k^2=m+n\\ \sqrt[3]{k^2+\cos^2 x}, & k^2<m+n\end{cases}$	$k=3.1;$ $m=5.15;$ $n=-0.5$
9	$y=\begin{cases}a\sin^{2.5}x+b\cos(zx+a), & x<a^3\\ (a+bx)^2-\sin(a+zx), & a^3\le x\le b\\ \sqrt{(\sin(a+bx+z)-x)}, & x>b\end{cases}$	$a=-2.2;$ $b=7.2;$ $z=e^x$

N вар	Формули для *y*	Параметри
10	$y=\begin{cases}\sqrt[3]{b^2+\sqrt{\lvert x+c\rvert}}, & \lg a+\lg b<\lg c\\ \cos(x-a+b-c), & \lg a+\lg b=\lg c\\ \sin(x+a-b+c), & \lg a+\lg b>\lg c\end{cases}$	$a=101;$ $b=9;$ $c=1112$
11	$y=\begin{cases}e^{ax}-3.5\cos^2(z+bx), & x\le a\\ a+\ln\lvert a+bx\rvert-2x, & a<x\le b^{3.5}\\ a+\cos^{3.5}(a+bxz), & x>b^{3.5}\end{cases}$	$a=-1;$ $b=3.4;$ $z=tgbx$
12	$y=\begin{cases}\ln\left(Lg\lvert kx+mn\rvert\right), & 3k>\lvert m+n\rvert\\ \sin kmx+\sqrt{\lvert nx\rvert}, & 3k=\lvert m+n\rvert\\ e^{kCosx}+e^{m+n}, & 3k<\lvert m+n\rvert\end{cases}$	$k=4;$ $m=-14.7;$ $n=-0.7$
13	$y=\begin{cases}x^2e^{2k}+\ln\lvert rx\rvert, & \cos k=\cos rs\\ \sqrt[3]{x^2}+\left\lvert\sqrt{k+rsx}\right\rvert, & \cos k>\cos rs\\ arctg(kx+rs), & \cos k<\cos rs\end{cases}$	$k=1.33;$ $r=0.85;$ $s=3,5$
14	$y=\begin{cases}2.5b^2+ax-4.5\cos xz, & x\le 5a\\ \left(a^2-5.4x\right)^3+\ln(xz), & x>b\\ \sqrt{6.5b^2+\left(a-x^3z\right)}, & 5a<x\le b\end{cases}$	$a=0.5;$ $b=4.5;$ $z=e^{ax}$
15	$y=\begin{cases}\sqrt{ax-\cos^2 b^3x+5.1c^2}, & \lvert 1-b^2\rvert=a+c\\ e^{0.007x}+\ln\lvert b^5\cos x\rvert, & \lvert 1-b^2\rvert>a+c\\ \cos^2 b^3x^2+\ln\lvert bx-a^2\rvert, & \lvert 1-b^2\rvert<a+c\end{cases}$	$a=3.5;$ $b=-0.73;$ $c=2.5$
16	$y=\begin{cases}3.5\sin(bx+z)-e^{3.5a}, & x\le a\\ \ln\left(a+b^3x\right)+a, & a<x\le b^{2.5}\\ \cos^2\left(a^b+xz\right)+a^2, & x>b^{2.5}\end{cases}$	$a=0.2;$ $b=0.5;$ $z=e^{2.5ax}$

N вар	Формули для *y*	Параметри
17	$y=\begin{cases} a+\sin bx+\cos x^2, & x<a \\ \sqrt{a+bx}+\sin zx, & a<x<\ln b \\ \ln(a+bx+z), & x>\ln b \end{cases}$	$a=0.2;$ $b=0.75;$ $z=\ln(bx)$
18	$y=\begin{cases} (3.5a-7.3bx+\sin zx), x<-\ln\lvert a\rvert \\ a^b-cos^3(a+zx), -\ln\lvert a\rvert \le x<b \\ \sqrt{\lvert \mathrm{tg}a-x\rvert}-x^2, & x>b \end{cases}$	$a=6;$ $b=3.2;$ $z=e^{1.5ax}$
19	$y=\begin{cases} c\sin b^2x+b\ln(cx+a), & x\le a \\ a+\ln(bx)-\sin^2(a+cx), a\le x<b \\ \sqrt{\cos(a+bx)+cx^2}, & x>b \end{cases}$	$a=2.2;$ $b=2.4;$ $c=\ln\lvert bx\rvert$
20	$y=\begin{cases} e^{ax}+f\cos^{3.5}bx, & x\le a \\ a+\cos^2 bx-\ln(fx), & a\le x<b^2 \\ \cos^2(a+bfx), & x>b^2 \end{cases}$	$a=0.8;$ $b=2.4;$ $f=e^{1.5ax}$
21	$y=\begin{cases} a\cos^2 x+b\sin zx, & x\le a \\ atg(ax+z)+\sin^2 bx, & a<x\le 4.5b \\ \ln(ax-b)+z^2, & x>4.5b \end{cases}$	$a=4.5;$ $b=8.4;$ $z=tgbx^2$
22	$y=\begin{cases} a+bx+\sin^2 zx^{3.5}, & x<a \\ a+\ln\lvert ab-zx^3\rvert+\ln x, & a\le x\le b^2 \\ \sqrt{\lvert a+tgzx\rvert}+b\sin x, & x>b^2 \end{cases}$	$a=0.3;$ $b=0.9;$ $z=\sin x^2$
23	$y=\begin{cases} \ln\lvert bzx\rvert+za^{2.5}, & a^5<x\le b \\ ax^2+bz^a+\sin^2 zx, & x>b \\ \cos(ax+b)+\ln\lvert zx\rvert, & x\le a^5 \end{cases}$	$a=1.5;$ $b=6.4;$ $z=\ln\lvert bx^3\rvert$
24	$y=\begin{cases} xe^x+(z+7.7abx), & x<a \\ tg(ax+z)+\cos^2 bx, & a\le x\le b^2 \\ \ln(\sin(a+bx+zx^2)), & x>b^2 \end{cases}$	$a=3.7;$ $b=8.7;$ $z=tgbx$

N вар	Формули для *y*	Параметри
25	$y=\begin{cases} a+z\cos^2 bx^3, & x<a \\ a+\sin^2 b^2+\ln(zx), & a<x<b \\ \sqrt[3]{0.3b+\sqrt{\lvert(a-z^2-\cos x)\rvert}}, & x>b \end{cases}$	$a = 1.5;$ $b = 5.7;$ $z = \ln\lvert tg\ bx\rvert$
26	$y=\begin{cases} c\cdot a^2+b\cdot\cos^3(a\cdot x), & a>x \\ \ln\lvert a\cdot c\cdot x\rvert+\sqrt{b}, & a\le xb \\ \cos(a+b\cdot x\cdot c+a\cdot c^2), & x>b \end{cases}$	$a=0.5;$ $b=6;$ $c=-5$
27	$y=\begin{cases} a^2+\sqrt{b^4+1.7}, & \lvert\cos x\rvert<0.2 \\ arctg(2^x-\lvert p\rvert), & \lvert\cos x\rvert=0.2 \\ \sqrt[3]{\ln\lvert a\rvert+4.3}, & \lvert\cos x\rvert>0.2 \end{cases}$	$a=0.5;$ $b=1.5;$ $p=-4$
28	$y=\begin{cases} a\cdot\sin^2 x+\ln\lvert b\cdot c\cdot x^2\rvert, & x<\frac{\pi}{2} \\ \sin^2(x^3+3a+\sqrt{b\cdot c}), & x=\frac{\pi}{2} \\ \cos(x/a)+b\sqrt{c}, & x>\frac{\pi}{2} \end{cases}$	$a=5;$ $b=5;$ $c=10$
29	$y=\begin{cases} a\cdot x-b^2+a\cdot\cos(c\cdot x), & x<a \\ c(a+b\cdot x)+\cos(b\cdot x), & x=a \\ a\cdot x^b+e^{c\cdot x}+\sin^2 x, & x>b \end{cases}$	$a=1{,}4$ $.b=1{,}7$ $c=3{,}7$
30	$y=\begin{cases} 7{,}1x^2+\frac{2{,}8\sin(c\cdot x)}{2a\cdot b}, & x\le a \\ \cos^2\left(\frac{2a}{3b}+c\cdot x\right), & a<x<b \\ \sqrt{\lvert x+a\cdot c^{0,5}\rvert}+\frac{a^2}{b}, & x\ge b \end{cases}$	a = - 5; b=5; c=2;

Завдання 4

Скласти схему алгоритму і програму обчислення суми членів ряду. Формулу для обчислення S взяти з табл. 9.

При розробленні програми використовувати оператор циклу з параметром (for).

Таблиця 4 – Варіанти індивідуальних завдань організації циклічних обчислень

1 $\sum_{k=1}^{7} \frac{(1)^k x^{2k}}{k!};\ x = 0,5$	**2** $\sum_{k=1}^{12} \frac{\cos^{k}(x)}{k};\ x = \frac{3\pi}{5}$	**3** $\sum_{k=11}^{9} \frac{x^{k}}{k!} Ln(x);$ $x = 0,84$
4 $\sum_{k=1}^{9} \frac{x^{k+1}}{(k+1)!}; x = 0,75$	**5** $\sum_{k=1}^{8} \frac{(-1)^k x^{2k}}{(2k)!}; x = 0,345$	**6** $\sum_{k=3}^{11} \frac{x^{2k}}{(2k-1)!}; x = 0,342$
7 $\sum_{k=1}^{12} \frac{Sin^{k}(x)}{k!}; x = 0,34$	**8** $\sum_{k=1}^{7} \frac{(xLn3)^k}{2^k}; x = 0,35$	**9** $\sum_{k=1}^{11} \frac{(x+1)^{2k}}{(2k)!};\ x = 0,71$
10 $\sum_{k=1}^{7} \frac{(-1)^k x^{k}}{2^{k} k!};\ x = 0,78$	**11** $\sum_{k=1}^{8} \frac{k^2 x^k}{(2k)!}; x = 1,21$	**12** $\sum_{k=2}^{10} \frac{x^{2k-1}}{2(k-1)!}; x = 0,84$
13 $\sum_{k=1}^{9} \frac{Sin(2kx)}{(2k)!}; x = 2,73$	**14** $\sum_{k=1}^{6} \frac{(-1)^k}{k^2} x^k; x = 0,92$	**15** $\sum_{k=1}^{12} x^k Cos^k\left(\frac{k}{4}\right); x = 1,51$
16 $\sum_{k=1}^{17} \frac{(-1)^k x^k}{k!}; x = 0,52$	**17** $\sum_{k=0}^{9} \frac{x^{k-2}}{(k+2)!}; x = 1,34$	**18** $\sum_{k=2}^{9} (-1)^k x^{2k}; x = 0,375$
19 $\sum_{k=1}^{10} k(2k+1)x^k; x = 0,19$	**20** $\sum_{k=1}^{11} \frac{kx^k}{4k^2-1}; x = 1,22$	**21** $\sum_{k=1}^{8} \frac{x^{2k-1}}{2^k(2k-1)}; x = 0,99$
22 $\sum_{k=0}^{16} \frac{(x-1)^k}{2k!}; x = 0,25$	**23** $\sum_{k=1}^{12} \frac{(x-1)^{2k}}{(x+1)^k}; x = -0,25$	**24** $\sum_{k=3}^{14} \frac{k^2 Cos(kx)}{(2k^2-1)}; x = 1,84$
25 $\sum_{k=2}^{9} \frac{(-1)^k x^{k-1}}{(k-1)!}; x = 1,92$	**26** $\sum_{k=1}^{10} \frac{x^{2k}}{2k(2k-1)}; x = 2,73$	**27** $\sum_{k=2}^{11} \frac{(k-1)x^k}{(k+1)3^k}; x = 1,12$
28 $\sum_{k=1}^{7} \frac{2x^{2k-1}}{3(2k-1)!}; x = 0,77$	**29** $\sum_{k=1}^{11} \frac{x^{2k+1}}{4k^2-1}; x = -0,96$	**30** $\sum_{k=2}^{9} \frac{x^{2k+1}}{(2k+1)2^k}; x = 2,75$

Завдання 5

Скласти схему алгоритму і програму обчислення таблиці значень функції ***y=f(x)***, при змінюванні значень ***x*** від ***xn*** до ***xk*** ($x \in$ [***xn, xk***]) з кроком h_x. Значення *a, b,* h_x та формули для обчислення значень ***y*** наведено в табл. 10.

Примітка. У програмі використати оператор циклу **while** або **do while**. Передбачити введення значень *a, b,* h_x з клавіатури і виведення здобутих результатів на форму у компонент Memo або ListBox.

Таблиця 10 – Варіанти для обчислення значень таблиць функції

№ вар.	Функція *y=f(x)*	Значення параметрів
1	$y=\begin{cases}6.3e^{-x}+\cos^{3.3}\left(ax+bx^2\right), & x^2 \le b\\ \ln\lvert ax^3+b\rvert-1.42x, & x^2 > b\end{cases}$	$a=-1.75;$ $b=3.28;\ h_x=0.4$ $x\in[0.4;\ 3.2]$
2	$y=\begin{cases}a^3\sin^3 x^2+e^a x-\sqrt{\lvert ex\rvert}, & ax \le \sqrt{\lvert x\rvert}\\ a^2 arctg\left(ax^3+b\right)+\cos^2 x, & ax > \sqrt{\lvert b\rvert}\end{cases}$	$a=0.71;$ $b=-17.5;\ h_x=0.9$ $x\in[3.5;9.8]$
3	$y=\begin{cases}\ln\lvert ax\rvert+b\cos^2 a^3 x-e^b, & x^2 \le \lvert a^{3.3}\rvert\\ \sqrt{\lvert 1.7x\rvert+2.8\lg\lvert bx\rvert}, & x^2 > \lvert a^{3.3}\rvert\end{cases}$	$a=-3.48;$ $b=-1.28;\ h_x=0.4$ $x\in[0.3;\ 10.1]$
4	$y=\begin{cases}\sin^2 a^2 x+\ln\lvert xb^2\rvert & x^2 < \sqrt{\lvert b\rvert}\\ e^{3a}-\sqrt{\lvert 0.77ax^3-\lg\lvert x\rvert\rvert}, & x^2 \ge \sqrt{\lvert b\rvert}\end{cases}$	$a=0.58;$ $b=-19.7;\ h_x=1.1$ $x\in[3.1;\ 8.8]$
5	$y=\begin{cases}\sqrt[3]{a^2 x}+bx^2-e^{0.05x}, & x^2 < 0.85a\\ \cos^2 b^3 x^2+\ln\lvert bx\rvert, & x^2 \ge 0.85a\end{cases}$	$a=2.47;$ $b=-0.9;\ h_x=0.5$ $x\in[0.2;\ 1.7]$
6	$y=\begin{cases}\sqrt{\lvert ax^2\rvert+\lvert\sin bx^{1.57}\rvert}, & 3x < \sqrt{b-a}\\ 1.4a^2 x-\ln\lvert ax\rvert+e^{b^2 x}, & 3x \ge \sqrt{b-a}\end{cases}$	$a=-1.38;$ $b=-0.6;\ h_x=0.15$ $x\in[0.3;\ 0.9]$
7	$y=\begin{cases}\sqrt[3]{\cos^2\left(a+bx^3\right)+5.1x}, & x < \sqrt{a}\\ e^{0.07x}+\ln\lvert b^5\cos x\rvert, & x \ge \sqrt{a}\end{cases}$	$a=2.81;$ $b=-3.67;\ h_x=0.2$ $x\in[0.7;\ 1.9]$

№ вар.	Функція $y=f(x)$	Значення параметрів
8	$y=\begin{cases}\sqrt{\lvert ax-\cos^2 b^3 x+\ln x^2\rvert}, & x^2 \ge \lvert a^3\rvert \\ \sin bx^{2.5}+e^{ax}, & x^2 < \lvert a^3\rvert\end{cases}$	$a=-1.98;$ $b=-0.75;\ h_x=0.3$ $x\in[0.3;\ 2.5]$
9	$y=\begin{cases}ax^2-e^{a^2x}+\ln bx^2, & x<\sqrt{9.8b} \\ \sqrt[3]{5.7b^2-a^2x}, & x\ge\sqrt{9.8b}\end{cases}$	$a=-3.8;$ $b=7.91;\ h_x=2.1$ $x\in[4.1;\ 10.4]$
10	$y=\begin{cases}\sqrt{\lvert abx\rvert+\sin^2 2bx}, & 2x>e^2 \\ \cos^2 x^3+\lg\lvert abx\rvert, & 2x\le e^2\end{cases}$	$a=3.8;$ $b=-2.5;\ h_x=0.3$ $x\in[1.5;\ 5.7]$
11	$y=\begin{cases}\sqrt[3]{10.31bx+e^{ax}}, & a>x^3 \\ \cos^2 bx+\ln ax, & a\le x^3\end{cases}$	$a=1.87;$ $b=-3.3;\ h_x=0.2$ $x\in[0.2;\ 1.4]$
12	$y=\begin{cases}4.11\ln bx+e^{ax}, & x^2\le b^{3.2} \\ tgax^2-\cos^2 a^2x, & x^2>b^{3.2}\end{cases}$	$a=1.8;$ $b=2.5;\ h_x=1.1$ $x\in[3.1;\ 8.6]$
13	$y=\begin{cases}e^{\lvert ax\rvert}+\sqrt{\lvert b^3x\rvert}, & \ln x<a \\ \sqrt[3]{arctgbx-4.7ax}, & \ln x\ge a\end{cases}$	$a=-0.355;$ $b=-4.3;\ h_x=0.1$ $x\in[0.5;\ 0.9]$
14	$y=\begin{cases}a^3+bx^2-\sin(a^3+bx^2), & x>\ln\dfrac{b}{2} \\ \sqrt{e^{ax}+10.7\lg bx}, & x\le\ln\dfrac{b}{2}\end{cases}$	$a=1.87;$ $b=11.3;\ h_x=0.1$ $x\in[1.1;\ 1.6]$
15	$y=\begin{cases}2.5\sin^2 b^2x+\lg ab^5x, & x<a^{1/2} \\ \sqrt[3]{e^{ax}-arctgb^2x}, & x\ge a^{1/2}\end{cases}$	$a=0.4;$ $b=2.57;\ h_x=0.15$ $x\in[0.1;\ 0.8]$
16	$y=\begin{cases}(a+\sin bx)(1.38-e^{-x}), & x\ge a \\ \sqrt[3]{ab^2x+arctgbx}, & x<a\end{cases}$	$a=7.31;$ $b=1.87;\ h_x=0.7$ $x\in[4.1;\ 10.4]$
17	$y=\begin{cases}\sqrt{27.4bx+e^{ax}}, & 1/x<a \\ arctgbx^2-\ln ax, & 1/x\ge a\end{cases}$	$a=1.4;$ $b=3.8;\ h_x=0.2$ $x\in[0.5;\ 1.7]$

№ вар.	Функція $y=f(x)$	Значення параметрів
18	$y=\begin{cases}\ln(11.5ax+33.1bx), & x>a+b\\ arctg^2ax-\cos x^3+b^2, & x\le a+b\end{cases}$	$a=0.48;$ $b=0.25;\ h_x=0.2$ $x\in[0.2;\ 1.0]$
19	$y=\begin{cases}e^{ax}+1.73b^4x^2, & x<0.25b\\ \sin^2 bx-\lvert\cos ax^2\rvert, & x\ge 0.25b\end{cases}$	$a=0.8;$ $b=2.4;\ h_x 0.2$ $x\in[0.1;\ 1.1]$
20	$y=\begin{cases}\sqrt{12.9ax+\ln bx}, & x\le a\\ arctga^2x-\lvert\sin bx^2\rvert, & a>x\end{cases}$	$a=2.49$ $b=8.28;\ h_x 0.9$ $x\in[3.5;\ 0.8]$
21	$y=\begin{cases}\sqrt[3]{arctg^2bx+\sqrt{e^{5ax}}}, & x<b\\ \sin b^2x-\ln(ax+5b), & x\le b\end{cases}$	$a=0.87;$ $b=1.44;\ h_x=0.5$ $x\in[0.3;\ 2.3]$
22	$y=\begin{cases}\lg\lvert bx^2\rvert+\cos^2 ax, & \ln a\ge x\\ \sqrt{e^2x+arctg^2ab^2x}, & \ln a<x\end{cases}$	$a=14.7;$ $b=-3.1;\ h_x=0.5$ $x\in[1.4;10.7]$
23	$y=\begin{cases}\sqrt[3]{1.411b^2x+\sin^2 ax}, & x<a^3\\ e^{x+b}+\ln\lvert bx\rvert\cdot\lg\lvert ax\rvert, & x\ge a^3\end{cases}$	$a=1.7;$ $b=-4.1;\ h_x=0.3$ $x\in[2.5;7.5]$
24	$y=\begin{cases}\ln ax\cdot e^{bx}+\sqrt{3.4b^2x}, & x<\cos a\\ 1.4x-arctgb^2x, & x\ge\cos a\end{cases}$	$a=0.5$ $b=-1.3;\ h_x=0.1$ $x\in[0.4;1.4]$
25	$y=\begin{cases}\sqrt[3]{3.77a^3x}+\sqrt{e^{-ax}}, & x>e^{\cos b}\\ \ln a^2x-\lvert\lg bx^2\rvert, & x\le e^{\cos b}\end{cases}$	$a=-4.71,$ $b=0.5;\ h_x=0.4$ $x\in[0.5;\ 8.5]$
26	$y=\begin{cases}4.45arctgax^2+7.1bx, & x\le\ln a\\ e^{a+b}-\lvert\sin^2 bx^3\rvert, & x>\ln a\end{cases}$	$a=7{,}77;$ $b=-4.4;\ h_x=0.3$ $x\in[0.3;\ 6.4]$
27	$y=\begin{cases}\sin\frac{3\pi}{2}x+\cos(5a+bx), & x\ge a^2\\ \lg a^5x-\sqrt{\lvert\cos bx\rvert}, & x<a^2\end{cases}$	$a=1.1;$ $b=0.74;\ h_x 0.25$ $x\in[o;\ 2.4]$

№ вар.	Функція *y=f(x)*	Значення параметрів
28	$y=\begin{cases}4.32\sqrt{\lvert abx\rvert+\lvert\cos x\rvert}, & \ln(a-b)\ge x\\ 17.3(x-b)-e^{-2ax}, & \ln(a-b)<x\end{cases}$	$a=3.4;$ $b=-3.3;\ h_x=0.4$ $x\in[0.1;\ 7.8]$
29	$y=\begin{cases}\sqrt{12.9ax+\ln bx}, & x\le a\\ arctga^2x-\lvert\sin bx^2\rvert, & a>x\end{cases}$	$a=2.18$ $b=3.28;\ h_x 0.4$ $x\in[0.4;\ 2.8]$
30	$y=\begin{cases}\ln bx+e^{ax}, & x^2\le b^{3.2}\\ ax^2-\cos^2 a^2x, & x^2>b^{3.2}\end{cases}$	$a=2.8;$ $b=2.5;\ h_x=1.1$ $x\in[3.1;\ 6.5]$

Завдання 6

Скласти схему алгоритму, розробити проект форми та програму алгоритмічною мовою С++ для опрацювання одновимірного масиву в середовищі С++Builder, відповідно до варіантів, наведених нижче.

В проекті передбачити введення елементів масиву з клавіатури через компонент Memo. При виконанні програми на комп'ютері значення елементів масиву (послідовність чисел) обрати самостійно.

Варіанти індивідуальних завдань

1. В масиві з 9 дійсних чисел знайти кількість елементів, які дорівнюють нулю.
2. Масив містить 12 дійсних чисел, знайти середньоарифметичне масиву.
3. Визначити кількість від'ємних елементів у масиві з 10 цілих чисел.
4. Масив містить 11 дійсних чисел. Знайти суму додатних елементів масиву.
5. Визначити мінімальне значення масиву, що містить 8 дійсних чисел.
6. В масиві з 10 цілих чисел замінити непарні за індексом елементи масиву на нулі.
7. Масив містить 12 дійсних чисел, знайти номер максимального елемента.
8. Обчислити загальну суму додатніх елементів двох масивів: масиву А із 5 цілих чисел та масиву В – із 6 дійсних чисел.
9. Упорядкувати за зростанням масив з 10 цілих чисел.
10. Масив містить 10 дійсних чисел. Сформувати новий масив, елементами якого є різниця між елементами заданого масиву та його середньоарифметичним.
11. Визначити індекс мінімального елемента масиву з 12 цілих чисел.
12. Масив містить 11 цілих чисел, знайти найменший з додатних елементів.

13.Визначити кількість елементів у масиві з 10 цілих чисел, які не перевищують значення 8.
14.Обчислити суму S додатних елементів одновимірного масиву з 13 цілих чисел та замінити непарні за значенням елементи на суму S.
15.В масиві з 11 дійсних чисел знайти кількість додатних, від'ємних та нульових елементів.
16.Обчислити добуток непарних елементів масиву з 11 цілих чисел.
17.Обчислити добуток додатних елементів одновимірного масиву з 10 цілих чисел та замінити парні за індексом елементи на добуток.
18.Масив містить 12 дійсних чисел, знайти мінімальний елемент масиву та поміняти його місцями з першим елементом.
19.В масиві з 9 цілих чисел знайти найменший з додатних елементів.
20.Обчислити суму елементів масиву з 14 цілих чисел, які не перевищують число 7.
21.В масиві з 10 цілих чисел знайти перший від'ємний елемент та поміняти місцями його з останнім елементом.
22.В масиві з 9 дійсних чисел знайти індекси максимального та мінімального елементів.
23.В масиві з 7 цілих чисел розташувати елементи масиву у зворотному порядку.
24.Обчислити суму елементів масиву з 11 цілих чисел, абсолютна величина яких не перевищує значення 5.
25.Визначити загальну кількість елементів у масиві з 10 цілих чисел, абсолютна величина яких більша за значення 3.
26.Масив містить 11 дійсних чисел, знайти добуток максимального елемента масиву на його перший елемент.
27.В масиві з 8 цілих чисел замінити всі елементи, які більші за число 5, на число –1.
28.Визначити кількість елементів у масиві з 12 дійсних чисел, які не перевищують значення середньоарифметичного.
29.Масив містить 9 дійсних чисел. Розташувати елементи масиву за зростанням значень їх модулів.
30.Масив містить 6 цілих чисел. Обчислити різницю між сумою елементів, індекси яких парні, та сумою елементів, індекси яких непарні.

Література

1. Буката Л.М., Кузнецов В.Д. Інформатика, модуль 1. Ч. 1. – Одесса: ОНАЗ, 2007.
2. Буката Л.М., Кузнецов В.Д., Ясинський В.В.- Інформатика, модуль 1.Ч. 2. – Одесса: ОНАЗ, 2008.
3. Трофименко О.Г. Інформатика, модуль 2. Ч. 1. – Одесса: ОНАЗ, 2007. – 60с.
4. Трофименко О.Г. Інформатика, модуль 2. Ч. 2. – Одесса: ОНАЗ, 2008. – 98с.
5. Шаповаленко В.А, Богатко К. А., Кузнецов В.Д. Швайко І.Г. , Єщенко І.А. Інформатика. Ч. 1. – Одесса: ОНАЗ, 2003.
6. Леонов Ю.Г., Угрік Л.М., Швайко І.Г. Збірник задач з програмування . – Одесса: УДАЗ, 1997.
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. Бобровский. Самоучитель программирования на языке С++ в системах Borland C++, Builder 4.0.
12. Бьерн Страуструп Язык программирования С++. – С.Пб.: М.: Бином, 1999 – 991 с.

Зміст