**Лабораторна робота №1**

**Програмування алгоритмів лінійної та розгалуженої структури**

**Мета роботи:** Ознайомитись із роботою середовища Visual Studio. Оволодіти навичками використання даного середовища програмування для створення програм на мові С++. Навчитись створювати програми на мові С++, в яких реалізуються лінійні алгоритми. Набуття навичок використання мови С++ для програмування з використанням операторів розгалуження.

**Короткі теоретичні відомості.**

Існують два стилі написання програм мовою С++: старий та новий. В ході розвитку мова С++ значно змінювалась. В результаті фактично існують дві версії мови C++. Перша з них — традиційна, розроблена Б’ярном Страуструпом. Друга версія розроблена Страуструпом і комітетом ANSI/ISO зі стандартизації. Вони дуже схожі, хоча друга версія має більші можливості. Якщо компілятор не підтримує новий стиль — можна писати в старому. Принципова різниця між старим та новим стилями полягає у двох властивостях: новий стиль підключення бібліотек і використання оператору namespace. Оголошення у директивах #include нового стилю не завжди є іменами файлів, тому у них не має розширення .h. Вони складаються лише з імені заголовку, що поміщене в кутові дужки. Єдина різниця між старим та новим стилем у заголовках є те, що заголовки нового стилю не обов'язково є іменами файлів. Оскільки мова С++ спадкувала всі бібліотеки функцій мови С, то підтримуються і стандартні бібліотеки мови С. (Тобто файли stdio.h, ctype.h і т.д. доступні) Але стандарт С++, крім них, визначає і заголовки нового виду, котрі можуть замінювати собою реальні файли. В програмах на мові С++ до імен стандартних файлів мови С додається префікс "с", а розширення ".h" відкидається.

Наприклад, файлу math.h в старому стилі відповідає заголовок <cmath>, а файлу string.h — заголовок <cstring>. Як правило, імена заголовків старого і нового стилю співпадають, за винятком розширення ".h". Всі компілятори дозволяють використовувати заголовкові файли старого стилю. Однак цей стиль вважається застарілим, і його не рекомендується використовувати в нових програмах. Простір імен (namespace) — це звичайна область видимості (область локалізації). Він призначений для локалізації імен ідентифікаторів і запобігає конфліктів між ними. Елементи, що оголошені в одному просторі імен, відокремлені від елементів, що належать іншому просторові. Якщо програма, що написана на мові С++, містить заголовок старого стилю, його вміст треба погрузити в глобальний простір імен: using namespace std;. Це дозволяє компілятору С++ компілювати програми, що написані на С. Приклад одної і тої самої простої програми, що написані у старому та новому стилях програмування: // Програма на С++ у старому стилі

#include <iostream.h>

int main()

{

return 0;

}

// Новий стиль програми мовою C++

#include <iostream>

using namespace std;

int main()

{

return 0;

}

Оператори введення-виведення <<, >>. Для роботи з цими операторами необхідно підключати заголовковий файл <iostream>. Відбувається робота з потоками введення-виведення.

**Структура найпростішої програми на мові C/С++.**

#include <iostream.h>

#include <conio.h>

int main() {

cout<< "Hello C++\n";

getch();

}

Модифікуємо нашу програму до вигляду:

#include <iostream.h>

#include <conio.h>

int main()

{

char name[20];

cout<< "What is your name:\n";

сіn >> name;

cout<< "Hello: " <<name << endl;

getch();

}

Після запуску вона повинна вивести на екрані питання What is your name:, ми відповідно повинні ввести ім’я, наприклад Ivan та натиснути клавішу **Enter**, на це програма повинна вивести: Hello: Ivan.

Продемонструємо ще один приклад найпростішої програми на мові C + +, яка запитує у користувача два цілочисельних значення змінних a і b, аналізує їх і виводить найбільше число (з використанням розгалуження).

#include <iostream.h>

#include <conio.h>

void main()

{

int a,b,max;

cout << "a="; //запрошення ввести значення a

cin >> a; //ввід значення змінної а

cout << "b="; // запрошення ввести значення b

cin >> b; // ввід значення змінної b

if(a>b) max=a; //якщо a>b то max=a

else max=b; //в протилежному випадку max=b

cout << "max="<<max; //вивід максимуму

getch();

} //зупинка до натискання клавіші

Перші два рядки підключають (include - включити) до тексту програми так звані заголовкові (h від header - заголовок) файли системи. У цих файлах описані системні функції і їхні аргументи. Використовуючи ці описи, компілятор перевіряє правильність виклику системних функцій. У нашому прикладі програма використовує системні функції введення (cin >>) і виводу (cout <<), опис яких знаходяться в заголовному файлі iostream.h, а також функцію очікування натискання будь-якої клавіші (getch), опис якої знаходиться в заголовному файлі conio.h. Назви заголовків файлів найчастіше утворюються від будь-яких абревіатур англійських слів, їх корисно навчитися розуміти, а не запам'ятовувати. У нашому прикладі: io - input / output (ввід / вивід), stream (потік), con (console - пульт оператора, тобто клавіатура і дисплей). Крім цих двох файлів ми будемо використовувати бібліотеки stdio.h для форматованого вводу та виводу даних та math.h – бібліотеку вбудованих математичних функцій.

Третій рядок містить заголовок функції main. Функція з такою назвою повинна бути в кожній програмі на мові C/C++. Саме з неї починається виконання програми, вона - головна (саме так перекладається службове слово main). Службове слово int (від integer - цілий) повідомляє, що результатом роботи функції main повинно бути ціле число, (за яким операційна система, запустивши програму main, може "вирішити", правильно чи неправильно завершилася робота програми). За загальноприйнятою угодою нульове значення, що повертається функцією main, свідчить про нормальне завершення роботи програми. Службове слово void (дослівно - порожнеча), зазначене в круглих дужках, повідомляє, що у функції main аргументи відсутні. Текст програми (тіло функції) розміщується в фігурних дужках ( 4-та 15-та стрічки). В п’ятій стрічці оголошено три змінні з іменами a, b і max, які можуть набувати цілочисельних значень. Окрім типу int в С/С + + можна використовувати і інші цілі типи даних, що відрізнятимуться діапазонами допустимих значень. Для десяткових дробів у найпростішому випадку використовуватимемо тип float (короткое дійсне, 4 байти) або double. Шоста стрічка є першою стрічкою програми, яка виконує якусь дію - вона виводить на дисплей повідомлення, що складається з двох символів (a =). Текст повідомлення обмежують подвійними лапками. Рядок 7 організовує зупинку роботи програми до тих пір, поки користувач не набере на клавіатурі якесь число і натисне клавішу Enter. Отримане значення буде сприйняте лише, якщо воно є цілим числом, і його буде направлено в змінну a. Подібним чином в рядках 8 і 9 буде організовано ввід значення числової змінної b. У десятій стрічці організовано порівняння поточних значень змінних a і b. Якщо значення змінної a більше, то воно присвоюється змінній max, в іншому випадку в змінну max заноситься значення b. Рядок 12 виводить на дисплей два повідомлення - текстове (max =) і числове (значення змінної max). Звернення до функції getch (рядок 13) призводить до затримки на екрані повідомлення програми до тих пір, поки користувач не натисне будь-яку клавішу (getch - від get character, дай символ).

**Завдання 1. Створення програми з реалізацією лінійного алгоритму**

1. Розглянемо створення програми для реалізації лінійного алгоритму на наступному прикладі.

Написати програму для розрахунку по двох формулах. Результати обчислень по двох формулах повинні співпадати. Для обчислень потрібно використовувати функції з математичної бібліотеки мови С++, які знаходяться у файлі **<cmath>**. Ввід вхідних даних здійснюється із клавіатури, а результати виводяться на екран.

Формули:

,

.

2. На початку програми підключимо необхідні файли:

#include <iostream>

#include <cmath>

using namespace std;

3. Із умови задачі випливає, що вводитись будуть два числа **m, n**. Реалізуємо програмно їх опис і ввід з клавіатури в тілі головної функції програми **main**:

void main()

{

double z1;

double z2;

double n;

double m;

cout << "n = "; cin >> n;

cout << "m = "; cin >> m;

…

}

Оператор **cout << "n = ";** здійснює виведення на екран підказки **n =** (пропонується ввести дійсне число). За ним слідує оператор введення **cin >> n;**, який вводить набране число в комп’ютер.

4. Далі обчислимо z1 і z2 по формулах, використовуючи математичні функції **sqrt** та **pow** з файлу **<cmath>**. Нові оператори добавляємо на місце три крапок.

z1=((m-1)\*sqrt(m)-(n-1)\*sqrt(n))/

(sqrt(pow(m,3)\*n)+n\*m+m\*m-m);

z2=(sqrt(m)-sqrt(n))/m;

5. Далі будуть слідувати оператори виводу результату:

cout << "z1 = " << z1 << endl;

cout << "z2 = " << z2 << endl;

Код програми завершується закритою фігурною дужкою, яка фактично закриває розділ операторів функції **main**.

6. Для компіляції і запуску програми виконуємо наступне: **Debug  Start Without Debugging,** або скористаємось комбінацією клавіш **Ctrl – F5**. Водимо додатні значення **n** і **m**. Останнє диктується областю визначення функцій z1 і z2 від аргументів *n* і *m*. Переконуємось в тому, що обчислені комп’ютером значення z1 і z2 співпадають.

7. Наведемо дані для тестування:

**n = 2**

**m = 3**

**z1 = 0,105946**

**z2 = 0,105946**

**Завдання 2. Створення лінійного алгоритму та написання відповідної програми**

Сторони трикутника рівні *a, b, c*. Скласти програму для обчислення площі трикутника S, радіуса вписаного кола *r*, та описаного кола *R*, його висот, медіан та внутрішніх кутів. Протестувати програму для вказаних значень *a, b, c*.



**Дані для тестування**:



**Короткі рекомендації при написанні програм**

1. Вибирайте тип змінних з урахуванням діапазону і необхідної точності представлення даних.
2. Давайте змінним імена, які відображають їх призначення.
3. Введення з клавіатури супроводжуйте запитом. Для контролю зразу ж після вводу даних виведіть вхідні дані на дисплей. Після завершення налагодження програми цей пункт можна вилучити з програми.
4. До запуску програми підготуйте тестові приклади, які містять необхідні дані і очікувані результати.
5. При записі виразів звертайте увагу на пріоритет операцій.
6. У функціях printf і scanf для кожної змінної вказуйте специфікацію формату, що відповідає її типу. Пам’ятайте, що в scanf передається адреса змінної, а не її значення.
7. При використанні стандартних функцій або класів потрібно за допомогою директиви #include підключити до програми відповідні заготовочні файли.
8. Не змішуйте в одній програмі ввід/вивід за допомогою класів ( в стилі С++) і за допомогою функцій бібліотеки (в стилі С).
9. Надавайте перевагу локальним змінним перед глобальними. Змінна повинна мати мінімальну із можливих областей дії.
10. Дані при вводі розділяйте пропусками, символами переводу рядка або табуляції.

**Протокол лабораторонї роботи**

Завершальним етапом виконання лабораторної роботи є оформлення протоколу лабораторної роботи. В заголовку протоколу потрібно вказати номер, назву та мету роботи. Далі потрібно описати хід роботи. Протоколюючи виконання індивідуального завдання обов’язково записати умову задачі, код програми з поясненнями окремих фрагментів, результати тестування, та результати моніторингу в середовищі **MathCad**. Протокол лабораторної роботи завершується висновками. Висновки повинні бути змістовними, з обговоренням проблемних ситуацій, які виникали при виконанні роботи.

**Програмування алгоритмів розгалуженої структури.**

Умовний оператор **if**. Даний оператор реалізує алгоритмічну структуру – розгалуження. Реалізується у двох варіантах:

* повне розгалуження: **if (P) S1; else S2;**
* неповне розгалуження: **if (P) S1;**

**P** – логічний вираз, приймає значення *true* або *false* (істинне або хибне). Дужки, у які береться логічний вираз, є обов’язковими. **S1** – оператр, виконується, якщо **P** – істинне**.** Оператор **S2** виконується у повному розгалуженні, якщо **P** – хибне.

Якщо у якісь із віток потрібно виконати кілька операторів, то їх потрібно заключити у блок {…}. У блоці можуть бути будь-які оператири в тому числі описи і інші умовні оператири, але блок не може складатись тільки з одних описів.

**Приклад 1.** Скласти програму для обчислення виразу:



Наведемо текст програми з використанням оператора **if** у форматі повного розгалуження:

#include <iostream>

#include <cmath>

using namespace std;

int main()

{

float x,y;

cout << "x="; cin >> x;

if (x>=0) y=sin(x); else

y=cos(x);

cout << "y=" << y << endl;

return 0;

}

Дані для тестування програми: для , для 

Зауважимо, що тригонометричні функції обчислюються від аргументів заданих у радіанах.

В даній програмі оператор **if** можна замінити на послідовність двох операторів **if** у форматі неповного розгалуження:

if (x>=0) y=sin(x);

if (x< 0) y=cos(x);

**Приклад 2.** Скласти програму для обчислення виразу:



Текст програми з використанням оператора **if** у форматі повного розгалуження:

**include <iostream>**

**using namespace std;**

**int main()**

**{**

**int x,y;**

**cout << "x="; cin >> x;**

**if (x<=0 || x>=1000) y=0; else**

**if (x>=1 && x<=9) y=1; else**

**if (x>=10 && x<=99) y=2; else y=3;**

**cout << "y=" << y << endl;**

**return 0;**

**}**

В даній програмі при формуванні логічних виразів використані конструкції з пар символів: **||** – або, **&&** - і.

**Приклад 3.** Знайти максимальне із трьох чисел x, y і z: :

Перепишемо умову задачі у наступному виді:



Програмна реалізація:

**#include <iostream>**

**using namespace std;**

**void main()**

**{**

**float x,y,z,u;**

**cout << "x="; cin >> x;**

**cout << "y="; cin >> y;**

**cout << "z="; cin >> z;**

**if (x>=y && x>=z) u=x; else**

**if (y>=x && y>=z) u=y; else u=z;**

**cout << "u=" << u << endl;**

**}**

**Приклад 4.** Скласти програму для визначення коренів квадратного рівняння



Дискримінант квадратного рівняння обчислюється за формулою

.

Далі розглядаються три випадки:

1. Якщо D>0, то рівняння має два розв’язки, які знаходяться за формулами



2. Якщо D=0, то рівняння має один розв’язок, який знаходиться за формулою



3. Якщо D<0, то рівняння немає розв’язків.

Таким чином початковими даними задачі є коефіцієнти *a, b, c*. Опишемо їх як дійсні змінні і введемо з клавіатури. Обчислимо *D*. Далі обчислювальний процес може піти по одній із трьох описаних вище віток. Скористаємось оператором **if** у форматі повного розгалуження:

#include <iostream>

#include <cmath>

using namespace std;

void main()

{

double a,b,c;

cout << "a="; cin >> a;

cout << "b="; cin >> b;

cout << "c="; cin >> c;

double d=b\*b-4\*a\*c;

if (d>0)

{

double x1=(-b-sqrt(d))/(2\*a);

double x2=(-b+sqrt(d))/(2\*a);

cout << "x1=" << x1 << endl;

cout << "x2=" << x2 << endl;

} else

if (d==0)

{

double x=-b/(2\*a);

cout << "x=" << x << endl;

} else

if (d<0) cout << "nema rozvjazku" << endl;

}

У даній програмі ми скористались вкладеними блоками {…} у першій та у другій вітках розгалуження.

Оператор **switch**. Оператор реалізує алгоритмічну структуру – вибір.

**switch (v)**

**{**

**case c1 : s1;**

**case c2 : s2;**

**........**

**case cn : sn;**

**default s;**

**}**

**v -** цілочисельний вираз,

**ci**- константний вираз.

Вихід з перемикача, як правило, здійснюється за допомогою **break** або **return**.

Всі константні вирази повинні мати різні значення, але повинні бути одного і того ж цілочисельного типу.

**Приклад 7.** Скласти програму – калькулятор.

**#include <iostream>**

**using namespace std;**

**void main()**

**{**

**float a,b,res;**

**char op;**

**cout << " 1-operand:"; cin >> a;**

**cout << " operacija:"; cin >> op;**

**cout << " 2-operand:"; cin >> b;**

**bool f=true;**

**switch (op)**

**{**

**case '+': res=a+b; break;**

**case '-': res=a-b; break;**

**case '\*': res=a\*b; break;**

**case '/': res=a/b; break;**

**default : cout << "error" << endl; f=false;**

**}**

**if (f) cout << "result:" << res << endl;**

**}**

**Самостійні завдання першого рівня**

|  |  |  |  |
| --- | --- | --- | --- |
| в | Завдання | в | Завдання |
| 1 |  | 2 |  |
| 3 |  | 4 |  |
| 5 |  | 6 |  |
| 7 |  | 8 |  |
| 9 |  | 10 |  |
| 11 |  | 12 |  |
| 13 |  | 14 |  |
| 15 |  | 16 |  |

**Індивідуальне завдання**

Студент в обовязковому порядку виконує одне завдання згідно варіанту. Прототипом індивідуального завдання є розглянуте вище завдання 2.

Наголосимо, що для довільних значень аргументів обчислені комп’ютером значення z1 і z2 повинні співпадати.

Обчислити z1 і z2 також у **MathCad**, і переконатись, що вони співпадають як між собою, так з відповідними результатами отриманими у Visual Studio.

У варіантах де у виразах фігурує константа , потрібно ініціалізувати константу, наприклад так: **double pi=3.1415926;**.

Фрагмент виразу, який починається із раціонального дробу, потрібно почати із дійсної константи. Наприклад фрагмент  доцільно запрограмувати так:

**z1 = 1./4-sin(5./2\*pi-a);**

Саме символ крапка приводить до сприйняття компіляторим чисельника як дійсного числа. І тоді результатом виконання фрагменту **1./4** буде число **0.25**. Якщо ж крапку опустити то результатом буде **0**, і результат обчислення всього виразу буде неправильними.

**Варіанти індивідуальних завдань**

1. 

2. 

3. 

4. 

5. 

6. 

7. 

8. 

9. 

10. 

11. 

12. 

13. 

14. 

15. 

16.

17. 

18. 

19. 

20. 

**Протокол лабораторонї роботи**

Завершальним етапом виконання лабораторної роботи є оформлення протоколу лабораторної роботи. В заголовку протоколу потрібно вказати номер, назву та мету роботи. Далі потрібно описати хід роботи. Протоколюючи виконання індивідуального завдання обов’язково записати умову задачі, код програми з поясненнями окремих фрагментів, результати тестування, та результати моніторингу в середовищі **MathCad**. Протокол лабораторної роботи завершується висновками. Висновки повинні бути змістовними, з обговоренням проблемних ситуацій, які виникали при виконанні роботи.

**Контрольні завдання**

1. У коді програми завдання 1 тип функції **main** замініть на **void**.
2. У коді програми завдання 1, в стилі С, тип змінної **i** замініть на **float**.
3. У коді програми завдання 2 замініть оператори введення виведення на відповідні функції у стилі С.
4. У коді програми завдання 2 локалізуйте опис змінних.
5. У коді програми завдання 2 оператори виведання z1 і z2 замініть одним оператором.
6. Складіть програму для перевірки головної тригонометричної тотожності.

**Контрольні питання**

1. Які типи даних вам відомо?
2. Які існують операції, та їхні приорітети?
3. Для чого потрібні функції **printf** i **scanf**?
4. Для чого призначена деректива **include**?
5. Чи можлива робота програми без **include**?
6. Як виділяється розділ операторів у програмі?
7. Який зв'язок між типом функції і оператором повернення з функції?
8. Які це глобальні і локальні змінні?
9. Що таке область дії змінної?
10. Які це прості і складені оператори просвоєння?