11 ЛЕКЦІЯ №11

Тема: Базові конструкції програмування в С/С++

ПЛАН

11.1 Загальні відомості
11.2 Оператори розгалуження
11.3 Оператори циклу
11.4 Оператори передачі керування

Час: 4 год.
Література:

11.1 Загальні відомості

Програму будь-якої складності можна скласти, використовуючи три базові структури програмування:

1) слідування;
2) розгалуження;
3) цикл;

Слідування – послідовне виконання двох і більш операторів.

Розгалуження задає виконання того чи іншого оператора в залежності від умови.

Цикл задає багаторазове виконання оператора.

Базові конструкції програмування можуть вкладатися друг у друга довільним образом, наприклад, усередині циклу може бути розгалуження.

Ціль використання базових конструкцій – одержати програму з простою і зрозумілою структурою. Такі програми легко читати, налагоджувати і вносити в них зміни.

У мові С/С++ реалізовані наступні конструкції програмування:

1) Прості оператори.
2) Оператор-вираження – будь-яке вираження, що завершується крапкою з ко-ми. Дія такого оператора полягає в обчисленні вираження.
   Приклади
   ```
   x++;
   a*=b;
   ```
3) Порожній оператор – оператор, що складається тільки із символу крапка з комою. Використовується там, де за синтаксисом потрібен оператор, але по логіці програми він повинний бути відсутнім.
4) Складений оператор – група операторів, укладених у фігурні дужки. Примітка – Кожен оператор завершується символом ; після закриваючої фігурної дужки крапку з комою не ставлять.

5) Блок – складений оператор, що містить визначення. Величини, визначені в блоці діють тільки в межах цього блоку.

Далі розглянемо оператори для організації нелінійних програм (розгалужених і циклічних).

11.2 Оператори розгалуження

До операторів розгалуження в C/C++ відносяться: умовний оператор `if` і перемикач `switch`.

Умовний оператор if.

Умовний оператор if використовується для розгалуження обчислювального процесу на два напрямки.

Повна форма оператора:

```
if (умова) оператор1; else оператор2;
```

Дія: Обчислюється вираження умови, якщо воно істинно, те виконується `оператор1`, в іншому випадку – `оператор2`;

Скорочена форма оператора:

```
if (умова) оператор1;
```

Дія: Обчислюється вираження умови, якщо воно істинно, те виконується `оператор1`, в іншому випадку керування передається наступному за `if` оператору.

Як умову можна використовувати арифметичне вираження або вказівник. Особливість умовного вираження: умова вважається істинною (`true`), якщо має ненульове значення, хибною (`false`) – якщо дорівнює нулю.

Приклади розгалужених програм:

Приклад 1. Визначити, є введене число парним чи непарним (як умову використовуємо таку ознаку: число буде парним, якщо залишок від ділення його на 2 дорівнює 0).

```
#include <iostream.h>
void main()
{
  int x;
  cout << "Увести x=";
  cin >> x;
  if (x%2==0) cout << "Число парне\n";
  else cout << "Число непарне\n";
}
```

Примітка – Тому що істинному значенню відповідає ненульовий результат умови, а хибному – нульовий, то рядок з оператором `if` можна записати так: `if (x%2)`.

Приклад 2. Програма для розв'язання квадратного рівняння.

```
#include <iostream.h>
#include <math.h>
void main()
{
  float a,b,c,d,x1,x2;
  cout << "Коефіцієнти=";
  cin >> a >> b >> c;
  d=b*b-4*a*c;
  if (d<0)
    {
      cout << "Немає коренів" << endl ;
    }
  else
    {
      x1=(-b+sqrt(d))/(2*a);
      x2=(-b-sqrt(d))/(2*a);
      cout << "x1=" << x1 << endl;
      cout << "x2=" << x2 << endl;
    }
}
```

Варіант програми рішення квадратного рівняння (для введення-виведення використовуються функції `scanf` і `printf`):

```
#include <stdio.h>
#include <math.h>
void main()
{
  float a,b,c,d,x1,x2;
  printf("Коефіцієнти=");
  scanf("%f%f%f",&a,&b,&c);
  d=b*b-4*a*c;
  if (d<0)
    {
      printf("Немає коренів\n");
    }
  else
    {
      x1=(-b+sqrt(d))/(2*a);
      x2=(-b-sqrt(d))/(2*a);
      printf("x1=%f\n",x1);
      printf("x2=%f\n",x2);
    }
}
```

Якщо потрібно перевірити кілька умов, то їх поєднують логічними операціями (&& – І, || – АБО).

Приклади складних умов:

`(0<x && 12<x)` – істинно, якщо *x* відповідає нерівності *0<x<12*

`(a=b || b=c)` – істинно, якщо істинна умова *a=b* або умова *b=c*

Оператор-перемикач switch

Призначений для розгалуження обчислювального процесу на декілька напрямків.

Формат оператора:

```
switch (вираження) {
     case константа1: оператори1;
     case константа2: оператори2;
     . . .
     case константа: оператори;
     default: оператори
}
```

Дія: Обчислюється вираження, потім його значення послідовно порівнюється з константами, що йдуть за `case`. При першому ж збігу виконуються оператори, що відповідають даній константі. Далі виконуються оператори всіх наступних варіантів, поки не з'явиться оператор переходу або не закінчиться перемикач. Оператори в розділі `default` виконуються, якщо значення вираження не збіглося з жодною константою після `case`. Розділ `default` може бути відсутнім, у цьому випадку при розбіжності значення вираження з константами, перемикач не виконує ніяких дій.

Таким чином, якщо значення вираження збіглося з константою номер К, те виконуються: `операториК`, `операториК+1`, `операториК+2` … `оператори`.

Примітки

1 Вираження і константи повинні бути символьного або цілочисельного типу.

2 В одному операторі `switch` не може бути двох однакових констант.

3 Щоб оператор `switch` міг працювати як альтернатива (аналогічно оператору `case` у Паскалі), необхідно, щоб кожна група операторів закінчувалася оператором `break`:

```
switch (вираження) {
     case константа1: оператори1;break;
     case константа2: оператори2;break;
     . . .
     case константа: оператори;break;
     default: оператори
}
```

Приклад:

Програма-калькулятор на 4 арифметичні дії. З клавіатури вводиться перше число, через пробіл – операція, потім через пробіл – друге число. Після натискання на Enter виводиться результат. Якщо введена неприпустима операція, то з'явиться відповідне повідомлення.

```
#include <iostream.h>
#include <conio.h>
void main()
{
 char op;
 int a,b;
 float x;
 clrscr();
 cout<<"Увести 2 числа та операцію:";
 cin >>a>>op>>b;
 switch (op) {
   case '+': x=a+b;break;
   case '-': x=a-b;break;
   case '*': x=a*b;break;
   case '/': x=a/b;break;
   default : cout<<"Неприпустима операція\n"; return;
 }
 cout <<"Результат="<<x;
}
```

11.3 Оператори циклу

Будь-який цикл складається з тіла циклу (операторів, що повинні виконуватися кілька разів), початкових установок, модифікації параметра і перевірки умови виконання.

У мові C/C++ є три види операторів циклу:

1) оператор циклу з передумовою;
2) оператор циклу з післяумовою;
3) оператор циклу з параметром;

Оператор циклу з передумовою

Формат оператора:

```
while (вираження) оператор;
```

Дія: Обчислюється значення вираження. Якщо воно істинно (не дорівнює нулю), виконується оператор (тіло циклу). Виконання продовжується доти, поки значення вираження не стане хибним (рівним нулю).

Приклад

Скласти таблицю значень функції $y=x^2$, якщо x змінюється від -2 до 10 із кроком 0.5.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  float x,y;
  clrscr();
  x=-2;
  while (x<=10)
  {
    y=x*x;
    cout<<"При x="<<x<<" \tзначення y="<<y<<endl;
    x+=0.5;
  }
}
```

Оператор циклу з післяумовою

Формат оператора:

```
do оператор while (вираження);
```

Дія: Виконується оператор (тіло циклу), потім обчислюється значення вираження. Якщо воно істинно (не дорівнює нулю), цикл виконується ще раз. Цикл завершиться, коли умова стане хибною (рівною нулю).

Приклад

Перевірка введення (потрібно, щоб було введено негативне число).

```
#include <iostream.h>
#include <conio.h>
void main()
{
  int x;
  clrscr();
  do
  {
    cout << "Увести негативне число=";
    cin >> x;
  }
  while (x>=0);
}
```

Оператор циклу з параметром

Формат оператора:

```
for (ініціалізація;вираження;модифікації) оператор;
```

де

`ініціалізація` – присвоєння початкових значень величинам, що використовуються у циклі.

вираження – визначає умову виконання циклу. Поки воно істинно, цикл виконується (фактично цикл із параметром реалізований як цикл із передумовою).

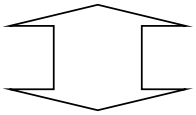
модифікації – виконуються після кожного проходу циклу і служать для зміни параметрів циклу.

Примітки

1 Кожна з частин оператора for може бути опущена, але крапки з комою повинні залишатися на своїх місцях.

2 Будь-який цикл while може бути приведений до еквівалентного циклу for і навпаки за наступною схемою:

```
for (ініціалізація;вираження;модифікації) оператор;
```



```
ініціалізація;
while (вираження);
{
  оператор;
  модифікації;
}
```

Приклад циклу з параметром

Скласти таблицю значень функції $y=x^2$, якщо x змінюється від -2 до 10 із кроком 0.5.

```
#include <iostream.h>
#include <conio.h>
void main()
{
  float x,y;
  clrscr();
  for (x=-2;x<=10;x+=0.5)
  {
    y=x*x;
    cout<<"При x="<<x<<" \tзначення y="<<y<<endl;
  }
}
```

11.4 Оператори передачі керування

У мові C/C++ існують наступні оператори, що змінюють природний порядок виконання:

- оператор безумовного переходу goto;
- оператор виходу break;

- оператор переходу до наступного ітерації циклу `continue`;
- оператор повернення з функції `return`.

Оператор goto

Формат:

```
goto мітка;
```

При цьому в тілі функції повинний бути рядок виду `мітка:оператор;` У якості мітки використовують довільний вільний ідентифікатор.

Оператор `goto` звичайно використовують у наступних випадках:

1) примусовий вихід з декількох вкладених циклів чи перемикачів;
2) перехід з декількох місць в одне, наприклад – на завершення функції.

Забороняється:

1) передавати керування усередину циклів, операторів `if` і `switch`;
2) переходити усередину блоків, що містять ініціалізацію змінних, минаючи ці рядки ініціалізації (значення змінних будуть невизначеними).

Оператор break

Використовується усередині операторів циклу або оператора switch для переходу в точку програми, розташовану за оператором, у якому знаходиться `break`.

Оператор continue

Дія: Пропускає всі оператори, що залишилися до кінця тіла циклу і передає керування в початок наступної ітерації.

Оператор return

Формат:

```
return вираження;
```

Дія: Завершує виконання функції. Вираження повинне мати скалярний тип. Якщо функція, у якій знаходиться `return` має тип `void`, то вираження повинне бути відсутнім (може бути відсутнім також сам оператор `return`).