Інструкції

- Види інструкцій C++
- Загальні зауваження щодо інструкцій
- Умовна інструкція `if`
- Перемикач `switch`
- Покроковий цикл `for`
- Зауваження щодо `for`
- Приклади використання циклу `for`
- Цикл з передумовою `while`
- Зауваження щодо `while`
- Цикл з післяумовою `do-while`
- Зауваження щодо `do-while`
- Прийоми використання циклів

Види інструкцій C++

Лінійні алгоритми

- Визначення (оголошення)
- `expr;` інструкція-вираз
- `{ }` складена інструкція
- `;` пуста інструкція

Галужені алгоритми

- `if` умовна інструкція
- `switch` перемикач
- `continue` інструкція продовження
- `break` інструкція виходу
- `return` інструкція повернення
- `goto` інструкція переходу

Циклічні алгоритми

- `for` покроковий цикл
- `while` цикл з передумовою
- `do-while` цикл з післяумовою

Загальні зауваження щодо інструкцій

- Усі інструкції, крім блоку, завершуються символом `;`
- Кожна інструкція може бути помічена міткою, яку розпізнає **`goto`**
- Обчислене значення **`expr`** може ігноруватися
- Цілочислові значення на місці **`bool`** неявно перетворюються до **`bool`**
- Інструкція-оголошення локалізує точку введення об'єкта в область видимості

Умовна інструкція `if`

```
if( expr ) instr1 ;

if( expr ) instr1 else instr2 ;
```

Алгоритм :

1. обчислюється **expr**
   і результат трактується (без приведення) як `bool`
2. якщо отриманий результат `true` , то виконується **instr1** ,
   інакше виконується **instr2** (якщо вона наявна)

!

- Якщо наявні **instr1 і instr2,** то виконається лише одна з них
- Якщо варіанти **instr1 і instr2** задають кілька дій, то їх оформляють у блок
- У повному `if` **instr1** завершується `;` (якщо не є блоком)
- Об'єкти, оголошені у **expr** , мають область дії до кінця **instr2**
- Конструкція `else` асоціюється з найближчою попередньою конструкцією `if`, яка явно не пов'язана з іншою `else`

Перемикач `switch`

```
switch( expr ){
    case const-expr1 : instr-list1
                       break;
    case const-expr2 : instr-list2
                       break;
     case const-exprk : instr-listk
                        break;
    default:  instr-list
}
```

Алгоритм перемикача

Алгоритм :

1. обчислення **expr**
2. співставлення отриманого результат зі значенням константних виразів у заданому порядку
3. при першому співпадінні значень виконання відповідних інструкцій, інакше виконання інструкцій *за замовчуванням* (якщо вони наявні)

!

- Якщо **`break`** (або **`return`**) відсутні серед інструкцій виконуваного варіанту, то після його завершення почнеться виконання інструкцій наступного варіанту
- Результат **expr** має бути інтегрального типу
- Має існувати неявне перетворення типу кожного константного виразу до типу **expr**
- Порядок **`case`**-варіантів є суттєвим
- Варіант **`default`** опційний(може бути відсутній)
- Об’єкти, оголошені у **expr** , мають область дії до кінця перемикача

C03:Menu2.cpp

```
int main() {
  bool quit = false;  // Flag for quitting
  while(quit == false) {
    cout << "Select a, b, c or q to quit: ";
    char response;
    cin >> response;
    switch(response) {
      case 'a' : cout << "you chose 'a'" << endl;
                 break;
      case 'b' : cout << "you chose 'b'" << endl;
                 break;
      case 'c' : cout << "you chose 'c'" << endl;
                 break;
      case 'q' : cout << "quitting menu" << endl;
                 quit = true;
                 break;
      default  : cout << "Please use a,b,c or q!"
                 << endl;
    }
  }
}
```

expr Type Promotion

```
   char response;
switch(int k=sizeof (response)) {
  case 1 : cout << "char:" <<k<< endl;   break;
  case 2 : cout << "short:"<<k<< endl;   break;
  case 4 : cout << "int:"  <<k<< endl;   break;
     default  : cout << "unexpected size" <<k<< endl;
   }

switch(int k=sizeof (response+1.0)) {
  case 1 : cout << "char:" <<k<< endl;   break;
  case 2 : cout << "short:"<<k<< endl;   break;
  case 4 : cout << "int:"  <<k<< endl;   break;
     default  : cout << "unexpected size" <<k<< endl;
 }
```

Покроковий цикл **for**

`for` **`(`** init-expr **`;`** cond-expr **`;`** loop-expr **`)`** instr **`;`**

Алгоритм :

1. Обчислення **init-expr**
2. Обчислення **cond-expr**
3. Якщо результат попереднього обчислення є **0**(трактується як **`false`**), то перехід на наступну після **`for`** інструкцію (завершення циклу)
4. інакше :
   - 4.1. виконання **instr**
   - 4.2. обчислення **loop-expr**
   - 4.3. перехід на **#2**

Зауваження щодо `for`

!

- **init-expr о**бчислюється лише 1 раз
- результат **init-expr** використовують (як правило) для ініціалізації *змінної циклу*
- ненульове значення **cond-expr** (`true`) є умовою виконання тіла циклу
- відсутність **cond-expr** еквівалентна `true`
- цикл з передумовою, тому **instr** може не виконуватися жодного разу
- **loop-expr** використовують для модифікації змінної циклу
- `for(;;){ ... }` - " вічний " цикл
- `break` або `return` під час виконання **instr** завершують виконання циклу
- `continue` під час виконання **instr** завершує поточну ітерацію і передає керування на #4.2.

Приклади використання циклу `for`

```
cout<<"minNumber?";
unsigned minNumber;
cin >> minNumber;
cout<<"maxNumber?";
unsigned maxNumber;
cin >> maxNumber;
```

```
//  послідовність квадратів натуральних чисел
//  від minNumber по maxNumber  включно

for( unsigned k=minNumber;  k <= maxNumber; ++k ){
   cout<<"k="<<k<<"  k2="<<k*k<<endl;
}
```

```
//  сума натуральних чисел
//  від minNumber по maxNumber  включно

unsigned s=0;
for (unsigned k=minNumber; k <= maxNumber; ++k) {
   s+=k;
}
cout<<"s="<<s<<endl;
```

```
unsigned minNumber=UINT_MAX-10;
unsigned maxNumber=UINT_MAX;
```

?

Цикл з передумовою `while`

```
while( expr ) instr ;
```

Алгоритм :

1. Обчислення **expr**
2. Якщо результат попереднього обчислення є **0**(трактується як `false`), то перехід на наступну після `while` інструкцію (завершення циклу)
3. інакше :
   - 3.1. виконання **instr**
   - 3.3. перехід на **#1**

Зауваження щодо `while`

!

- **expr о**бчислюється на початку кожної ітерації
- ненульове значення **expr** (`true`) є умовою виконання тіла циклу
- наявність **expr** обов'язкова
- цикл з передумовою, тому **instr** може не виконуватися жодного разу
- під час виконання **instr** (або **о**бчислення **expr)** компоненти **expr** мають модифікуватися – інакше " вічний " цикл
- `break` або `return` під час виконання **instr** завершують виконання циклу
- `continue` під час виконання **instr** завершує поточну ітерацію і передає керування на #1.

Цикл з післяумовою `do-while`

```
do  instr  ; while(  expr  );
```

Алгоритм :

1. Виконання **instr**
2. Обчислення **expr**
3. Якщо результат попереднього обчислення є **0** (трактується як `false`), то перехід на наступну після `do-while` інструкцію (завершення циклу)
4. інакше перехід на **#1**

Зауваження щодо `do-while`

!

- **expr о**бчислюється після виконання **instr**
- ненульове значення **expr** (`true`) є умовою виконання тіла циклу
- наявність **expr** обов'язкова
- цикл з післяумовою, тому **instr** буде виконуватися принаймні 1 раз
- під час виконання **instr** (або **о**бчислення **expr)** компоненти **expr** мають модифікуватися – інакше " вічний " цикл
- `break` або `return` під час виконання **instr** завершують виконання циклу
- `continue` під час виконання **instr** завершує поточну ітерацію і передає керування на #2.

Прийоми використання циклів

1. **Вибір виду інструкції циклу**
   - **`for`** - відома наперед к-сть ітерацій
   - **`while`** - к-сть ітерацій наперед невідома, їх може бути 0
   - **`do_while`** - відбудеться принаймні 1 ітерація, але їх к-сть наперед невідома
2. **Очевидний спосіб керування ітераціями**
   - **`break`** - достроковий вихід з циклу
   - **`continue`** - завершення поточної ітерації
3. **Код ініціалізації розміщувати безпосередньо**
   - перед інструкцією **`while`** або **`do_while`**
   - у заголовку **`for`**

Прийоми використання циклів

4. Службові інструкції керування циклом групувати в одному місці
5. Вирази в умові повторення ітерацій мають бути простими
6. Керуючі змінні циклу не використовувати не за призначенням
7. Глибина вкладення циклів <=3