Лабораторна робота № 6

Тема: Програми з повтореннями (цикли while; do-while).

Мета: Набути навичок написання програм з циклами while; do-while.

Теоретичні відомості

Цикл while

Цикл **while** зручно застосовувати у випадку, якщо в програмі необхідно повторювати дії, поки виконується якась умова. Заздалегідь ми не знаємо, коли вона перестане виконуватися й скільки разів виконається, відповідно, цикл. Наприклад, щоб змоделювати процес завантаження автомобіля за допомогою екскаватора, потрібно послідовно уводити значення маси чергової порції піску і перевіряти, чи не перевищена вантажопідйомність автомобіля. Кількість ітерацій такого циклу невідома, оскільки маса піску в ковші екскаватора щоразу інша, тому слід використати цикл з умовою.

Існує 2 форми написання циклу **while:**

```
while(умова)
 оператор_тіла_циклу;
```

```
do
 оператор_тіла_циклу;
while(умова);
```

У першому випадку, спочатку проводиться перевірка умови, і якщо вона істинна – виконуються оператори, що складають тіло циклу. Цикл буде виконуватися, поки умова залишатиметься істинною. При першому ж порушенні істинності умови цикл припиниться. У такій формі запису циклу **while** можлива ситуація, коли тіло циклу взагалі не буде виконане: якщо умова при першій перевірці виявиться хибною.

У другій формі запису тіло циклу обов’язково буде виконане хоча б один раз, тому що умова перевіряється після першого виконання операторів тіла циклу.

Як у першій, так і в другій формі оператор тіла циклу може бути складеним:

```
while(умова)
 {оператори}
```

```
do
 {оператори}
while(умова);
```

Вдалий вибір виду циклу **(for, while** чи **do...while)** робить програму зрозумілішою, а отже зменшує ймовірність помилки. Розглянемо приклади.

Алгоритм Евкліда. Алгоритм знаходження найбільшого спільного дільника двох натуральних чисел **m** і **n** (НСД(m,n)) був описаний в III столітті до н.е. в класичному трактаті «Початки» грецького математика Евкліда.

Розв'язок можна одержати шляхом послідовного віднімання меншого числа від більшого доти, поки m≠n:

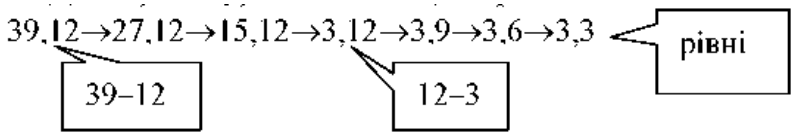
крок 1: порівняти m і n; якщо m=n, то НСД(m,n)=n, інакше крок 2;

крок 2: визначити більше з чисел;

крок 3: відняти від більшого числа менше. Отриманою різницею замінити більше число; крок

4: перейти до кроку 1.

Для пари натуральних чисел, наприклад 39 і 12, це виглядає так:

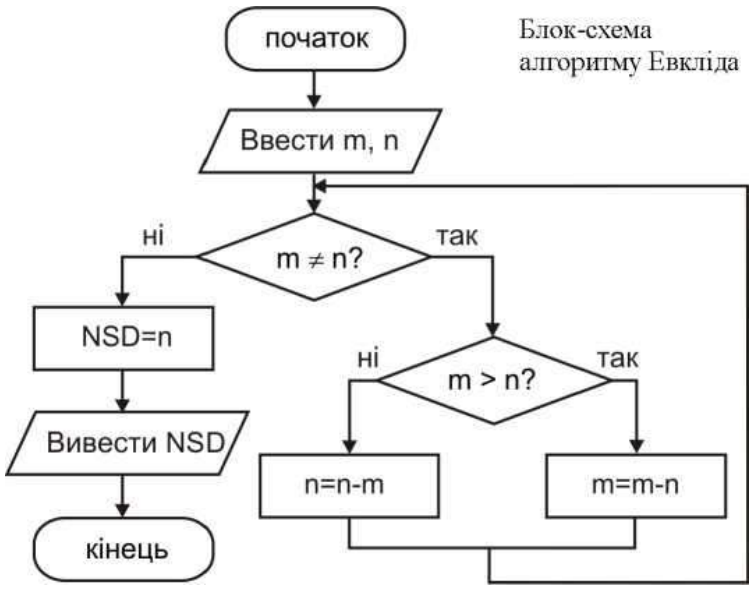


Зі схеми видно, що НСД(39,12)=3. Використаємо відомі нам оператори C++:

поки числа не стануть рівними,	`while(m!=n)`

більше число замінюватимемо різницею більшого і меншого	`if(m>n)`
	`  m=m-n;`
	`else`
	`  n=n-m;`

У результаті одержимо пару однакових чисел **m** і **n,** які й дорівнюють найбільшому спільному дільнику.



Блок-схема алгоритму Евкліда

```
#include<iostream.h>                    //Програма 3.5
#include<conio.h>
int main()
{
int m,n,NSD;
cout<<"Vvedi m,n:";cin>>m>>n;
   while(m!=n)
   {
     if(m>n)
       m=m-n;
     else
       n=n-m;
   }
   NSD=n;
   cout<<"NSD="<<NSD<<endl;
getch();
return 0;
}
```

***Приклад** 2. Увести послідовність додатних та від'ємних чисел, що закінчується нулем, і визначити суму додатних:*

```
#include<iostream.h>                    //Програма 3.6
#include<conio.h>
int main()

{
int k, S=0;
   do
   {
      cout<<"Vvedi k:";cin>>k;   //уведення даних
      if(k>0)                   //відбір додатних
            S=S+k;               //підсумовування
   }
   while(k!=0);            //перевірка закінчення
cout<<"Summa dodatnih S="<<S<<endl;
getch();
return 0;
}
```

У цій програмі цикл використовується як для підсумовування даних, так і для перевірки закінчення уведення даних (при **k**=0).

Приклад 3. *Підрахувати* **k** *- кількість цифр у десятковому записі цілого невід'ємного числа* ***n***.

Поділимо шукане число націло на 10, потім поділимо отриманий цілий результат на 10, і так доти, поки не одержимо в результаті нуль. При цьому змінна-лічильник **k,** збільшуючись після кожного ділення, лічить кількість ітерацій, яка й дорівнює розрядності числа **x.** У даній програмі для вирішення поставленого завдання цикл повинен виконатися хоча б один раз, тому скористаємось формою **do...while.**

```
#include<iostream.h>                    //Програма 3.7
#include<conio.h>
int main()
{
int n;                                          //число
int k=0;                                      //лічильник
cout<<"Vvedi cile dodatne chislo:";cin>>n;
int ost=n;                             //цілий результат
   do
   {
     ost=ost/10;
     k=k+1;
   }while(ost!=0);
cout<<k<<"-cifrove"<<endl;
getch();return 0;
}
```

<u>Вкладені цикли</u>

Цикли можуть бути вкладеними один в одний, тобто тіло циклу може включати в собі оператор циклу. Кількість вкладених циклів (ще кажуть: «глибина вкладення») не обмежена.

Приклад. *Вивести на екран прямокутник, заповнений символами «*»:*

```
********
********
******** і т.д.
```

```
#include<iostream.h>                    // Програма 3.8
#include<conio.h>
int main()
{
int i,j,n,m;
cout<<"Rjadkiv: ";cin>>n;
cout<<"Stovpciv: ";cin>>m;
for (i = 0; i < n; i++)
{
   for (j = 0; j < m; j++)
     cout<<"*";
  cout<<endl;
}
getch();return 0;
}
```

У цій програмі зовнішній цикл використовує змінну циклу **i**, а внутрішній - змінну циклу **j.** Для кожного значення і внутрішній цикл виконується m разів. Таким чином, оператор виведення буде друкувати у рядку m символів «*», а потім, при переході до наступного значення змінної **i**, завдяки оператору **cout<<endl;** буде здійснюватись перехід на новий рядок. Це повторюватиметься доки, поки змінна зовнішнього циклу не досягне значення **n.**

Щоб не заплутатися у вкладених циклах, обов'язково використовуйте табуляцію при написанні програми! Це зробить вашу програму зрозумілішою і спростіть пошук помилок.

Хід роботи

Завдання 1.

1. Випробуйте програму 3.5. для m=39, n=12. Доповніть її так, щоб були виведені проміжні результати обчислень (як, наприклад, у програмі 3.3).

2. Випробуйте програми 3.6. та 3.7. Чому при введенні більше ніж 10-цифрових чисел програма 3.7. працює неправильно?

3. Випробуйте програму 3.8. Виведіть замість символу «*» своє ім'я.

Завдання 2.

1. На дверях ліфта висіло загрозливе попередження про те, що двері зачиняються самі в той момент, коли зайвий за вагою пасажир переступить поріг ліфта. Який за рахунком пасажир спричинить аварію, якщо ліфт витримує

вагу не більше S кг, а вага кожного з n пасажирів, що стоять у черзі до ліфта, дорівнює, відповідно, a_1 a_2, ..., a_n?

2. Обчислити факторіал цілого числа, уведеного з клавіатури. Дослідити, в якому діапазоні вхідних даних програма працює правильно.

*Примітка. Факторіалом числа n називається добуток усіх натуральних чисел від 1 до n. Записується це так: n!=1*2*3...*n. Наприклад:* ***4!=1*2*3*4=24.***

3. Увести послідовність чисел, що закінчується нулем, і визначити найбільше число в ній.

Завдання 3.

1. Вивести на екран заповнений символами «*» трикутник:

```
*
**
***
і т.д.
```

2. Надрукувати на екрані таблицю Піфагора.

3. Задані 3 цілих додатних числа: a, b, c. Визначити, чи можна з відрізків з такими довжинами утворити трикутник.

4. Збережіть програми у власній папці з назвою лабораторної роботи.

Контрольні запитання:

1. У яких випадках зручно використовувати цикл while?
2. Назвіть дві форми використання циклу while?
3. Виконайте усно алгоритм Евкліда для пари чисел 24 і 16?
4. Для чого використовують змінну-лічильник?
5. Чи може використовуватися цикл у якості оператора в тілі іншого циклу?
6. Для чого використовують табуляцію при написанні програм?