Оператор циклу з передумовою while

Оператор while використовується для організації циклічного виконання дій, поки виконується певна умова.

while (<логічний вираз>) { <тіло циклу> }

Приклад 1: Обчислити суму чисел от 1 до 100.

```
#include <stdio.h>
int s,i;
void main(void)
{
  s = 0; i = 100;
  while(i > 0)
  {       s = s + i;
            i--;

   }
  printf("%d\n",s);
}
```

Тіло циклу – довільний оператор (зокрема блок).

Оператором ***while*** виконуються наступні дії:

1) обчислюється значення виразу;

2) якщо значення виразу дорівнює нулю (умова хибна), то виконання циклу завершується і керування передається оператору, наступному за ***while***;

3) якщо значення виразу ненульове (умова істинна), то виконується оператор тіла циклу;

4) відбувається повернення до п.1 для наступної перевірки умови виконання циклу.

Приклад 2: Обчислити **y=sin(x)+k**, де **x** змінюється від **xmin** до **xmax**

з кроком **x**.

```
#include<stdio.h>
#include<math.h>
void main( )
{   double x, y, xmin, dx, xmax, k=0.25;
     puts (“Введіть  xmin, dx, xmax”);
    scanf (“%lf%lf%lf”, &xmin, &dx, &xmax);
    x=xmin;
    while (x<=xmax)
       {     y=sin(x)+k;
             printf (“При x=%lf y=%lf\n”, x, y);
            x+=dx;     }
  puts (“Вітаю з успіхом!”); }
```

Оператор циклу з постумовою do … while

Оператор ***do…while*** використовується для організації циклічного виконання оператора або серії операторів, які називаються тілом циклу, до тих пір, поки умова не стане хибною.

Синтаксис : ***do { <оператор>; }***

while (<логічний_вираз>);

Цикл обов’язково виконується (хоч один раз).

Використання оператора ***do … while*** зручне у випадках, коли значення виразу-умови залежить від результатів виконання тіла циклу.

Приклад : Обчислити суму чисел от 1 до 100.

```
#include <stdio.h>
int s,i;
void main(void)
{
  s = 0; i = 100;
  do {
    s = s + i;
    i--;
  } while (i > 0);
  printf ("%d\n", s);
}
```

Цикл ***do ... while*** припиняє виконання, коли умовний вираз обертається в нуль (стає невірним).

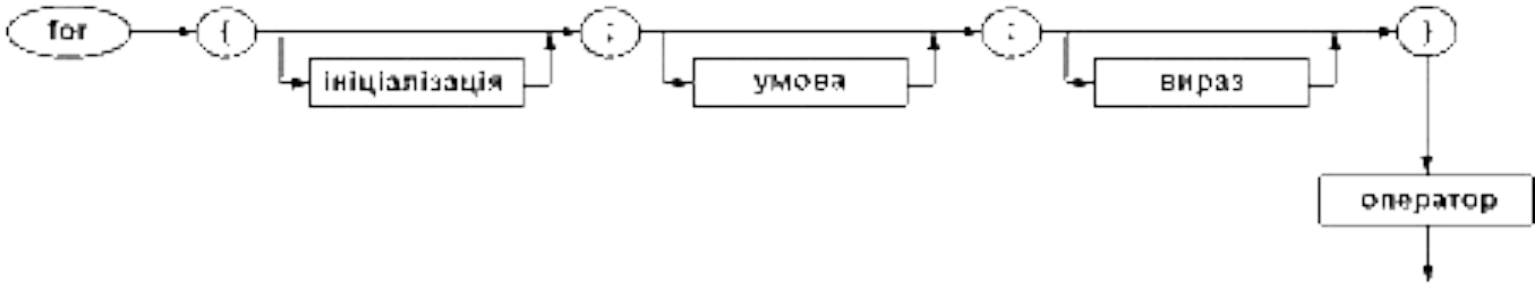
Оператор циклу for

Оператор ***for*** забезпечує циклічне повторення деякого оператора певне число разів. Оператор, який повторюється називається тілом циклу. Повторення циклу звичайно здійснюється з використанням деякої змінної (лічильника), яка змінюється при кожному виконанні тіла циклу. Повторення завершується, коли лічильник досягає заданого значення. Синтаксис оператора:

for ([ініціалізація]; [перевірка_умови];[нове_значення])

{ оператор ;

}



Формально роботу циклу можна описати такими кроками

1)якщо перший вираз (ініціалізація) присутній, то він обчислюється;

2) обчислюється вираз умови (якщо він присутній). Якщо умова дорівнює 0, тобто вона невірна, цикл припиняється, у протилежному випадку він буде продовжений;

3) виконується тіло циклу;

4) якщо присутній вираз зміни лічильника, то він обчислюється;

5) надалі перехід до пункту під номером 2.

Поява у будь-якому місці циклу оператора ***continue*** призведе до негайного переходу до пункту 4.

Приклад : обчислити суму чисел от 1 до 100

```
for (s=0, i=1; i<=100; i++)
s = s + i;
```

Приклад: друк парних чисел у проміжку від 500 до 0

```
#include <stdio.h>
    void main(void)
    {
        long i;
        for (i=500; i>=0; i-=2)
        printf ("\n%ld", i);
        printf ("\n");
    }
```

Для того, щоб продемонструвати гнучкість даного різновиду циклу, представимо весь перелік обчислень лише в одному операторі ***for***, за яким слідує порожній оператор:

```
#include <stdio.h>
void main(void)
{
    long i;
    for (i=500; i>=0; printf ("\n%ld", i), i-=2) ;
}
```

Вкладення циклів

Якщо у тілі циклу організовано внутрішні циклічні дії, то такі цикли називають вкладеними. Приклад – виведення на екран піраміди літер.

```
#include<stdio.h>
#define rmax 8          /*висота піраміди*/
int main()
{
  int r, left, right, pos;  /*номери рядка,початку і кінця рядка*/
  for (r=1, left=right=rmax; r<=rmax; left--, right++, r++)
   {  for (pos=1; pos<left; pos++)
      putchar(‘ ‘);              /*виведення символу пробілу*/
      for ( ; pos<=right; pos++)
     putchar(‘A’+r-1); /*виведення літер рядка r з кодом
                                 на r-1 більшим за ‘A’*/
   putchar (‘\n’);
    }
  return 0;
}
```

Результат виконання програми

A

BBB

CCCCC

DDDDDDD

EEEEEEEEE

FFFFFFFFFFF

GGGGGGGGGGGGG

HHHHHHHHHHHHHHH

Оператор продовження continue

Оператор continue передає управління на наступну ітерацію в операторах циклу ***do, for, while***. Він може розміщуватися тільки в тілі цих операторів. В операторах ***do*** і ***while*** наступна ітерація починається з обчислення виразу умови. Для оператора ***for*** наступна ітерація починається з обчислення виразу зміни значення лічильника.

Приклад :

```
while (i-- > 0)
{
        x=f (i);
        if (x == 1) continue;
        else y=x*x;
}
```

Оператор return

Завершує роботу функції, в якій він виконується. Керування програмою повертається до оператора, з якого була викликана ця функція.

return **-** без повернення значення для функцій, що мають тип ***void*** ;

return *вираз* - повертає значення вказаного виразу (після перетворення до типу функції) в точку виклику фунуції.

Виконання оператора ***return*** всередені ***main()*** завершує роботу програми.

Використання псевдовипадкових чисел

Генерування послідовностей рівномірно розподілених випадкових чисел.

Потрібно підключити бібліотеку ***#include <stdlib.h>***

Функція ***int rand (void)*** повертає псевдовипадкове число з проміжку ***0 ..INT_MAX*** (0,,32767)

Функція ***void srand (unsigned seed)*** змінює стартове значення функції ***rand ,*** щоб у разі повторного запуску програми генерувалась інша послідовність чисел. Параметр функції – число, що слугуватиме затравкою для наступного виклику ***rand .***

У бібліотеці Borland C є додаткові функції:

int random (int hval) - повертає псевдовипадкове число з діапазону *0 .. hval-1 ;*

void randomize (void) - для запуску генератора з поточним значенням системного таймера.

Приклад – формування цілого випадкового числа з проміжку RMIN .. RMAX

```
.....
int numb;
randomize ();
numb = RMIN + random (RMAX – RMIN +1) ;
.....
```