

# *Етапи розв'язання задач на комп'ютері*

**Постановка** задачі мовою чітко визначених математичних понять. Суть поставленої задачі, необхідні початкові дані та інформацію, що вважається результатами розв'язання.

**Побудова** математичної моделі. Створюється інформаційна математична модель об'єкта, і чим достовірніше вона відображає реальні сторони об'єкта, тим точніші одержані результати.

**Розробка** алгоритму, тобто послідовності вказівок для розв'язання задачі, відбувається на основі побудованої математичної моделі. При створенні складних алгоритмів застосовується метод покрокової розробки, сутність якого полягає в тому, що алгоритм розробляється «зверху донизу»: необхідно розбити алгоритм на окремі частини, кожна з яких розв'язує свою самостійну підзадачу, і об'єднати ці підзадачі в єдине ціле.

**Складання програми.** Алгоритм має бути записаний мовою програмування. Може здійснюватися теж за принципом «зверху донизу», що дозволяє одержати добре структуровану програму, читання і розуміння якої значно полегшене.

**Компіляція програми.**

**Компонування програми.**

**Налагодження програми.** Полягає в підготовці системи тестів, які містять набір вихідних даних, що мають відомий результат.

**Експлуатація програми.** Програма, що має відповідну документацію, може бути тиражована і запропонована іншим користувачам.

# Оператори

**Оператори** - це основні елементи, з яких "будуються" програми, призначені для виконання встановлених дій.

За конструкцією оператори поділяють на групи: *прості*, *складені*.

За характером дій: оператори-вирази, умовні оператори, переходу, циклу.

Окремий вид складеного оператора – *блок*. Це група довільних операторів, об'єднаних фігурними дужками {...}.  
Всередині блоку можна оголошувати локальні змінні.

```
{  
double tm;  
    tm=u; u=v; v=tm;  
}
```

# Оператори-вирази

Кожен допустимий вираз, що закінчується ;

*clrscr(); z=3.5\*x;*

Виділяють: оператори присвоєння та оператори звертання до функцій.

# Умовні оператори

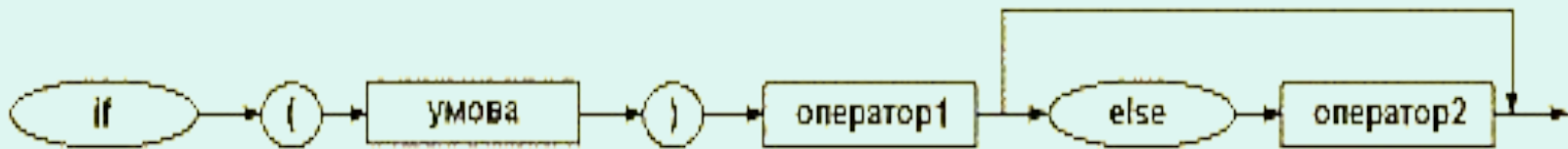
Реалізують розгалуження процесу виконання програми.

Використовують два види : *if* , *switch*

# Оператор розгалуження *if*

Оператор розгалуження призначений для виконання тих або інших дій в залежності від істинності або хибності деякої умови.

Синтаксис оператора : ***if*** (<умова>) <оператор1>;  
***[else*** <оператор2;>***]***



Умова хибна, якщо вона дорівнює нулю, в інших випадках вона істинна

Скорочений запис ***if*** (вираз) оператор;

Вкладення умовних операторів. За правилами – кожна ***else***-частина умовного оператора пов'язується з найближчим ***if*** (інакше використовуємо фігурні дужки).

## Приклад 1

*/\* програма виводить результат ділення двох дійсних чисел \*/*

```
#include<stdio.h>
```

```
void main()
```

```
{ float a,b,c;
```

```
printf ("Введіть число a :\n");
```

```
scanf ("%f",&a);
```

```
printf ("Введіть число b :\n");
```

```
scanf ("%f",&b);
```

```
if (b==0) printf ("Ділення да нуль !\n");
```

```
else
```

```
{ c=a/b;
```

```
printf ("a : b == %g",c);
```

```
}
```

```
}
```

## Приклад 2

Обчислити значення функції:

$$y(x) = \begin{cases} x + 1, & x < 0 \\ x^2, & 0 \leq x < 10 \\ x - 4, & x \geq 10 \end{cases}$$

```
#include <stdio.h>
```

```
double x,y;
```

```
void main(void)
```

```
{
```

```
scanf ("%f",&x);
```

```
if (x < 0) y = x + 1; else
```

```
if (x < 10) y = x*x; else y = x - 4;
```

```
printf ("%f\n",y);
```

```
}
```

# Оператор *break*

Призначений для переривання роботи оператора вибору і операторів циклу.

Перериває виконання внутрішніх операторів *switch* і передає керування оператору, наступному за оператором вибору.

Якщо оператор *break* викликається в тілі циклу, то виконання циклу відразу припиняється і керування переходить до оператора, наступного за оператором циклу



# Оператор *switch*

Синтаксис :

```
switch(<вираз цілого типу>
{
  case <значення_1>:
    <послідовність_операторів_1>;
  break;
  case <значення_2>:
    <послідовність_операторів_2>;
  break;
  .....
  case <значення_n>:
    <послідовність_операторів_n>;
  break;
  [default:
    <послідовність_операторів_n+1>;]
}
```

Приклад :

```
switch(i) {  
case -1: n++;  
break;  
case 0: z++;  
break;  
case 1: p++;  
break;  
}
```

За відсутності операторів **break** відбувається послідовне виконання всіх внутрішніх операторів, починаючи з вибраної гілки розгалуження.