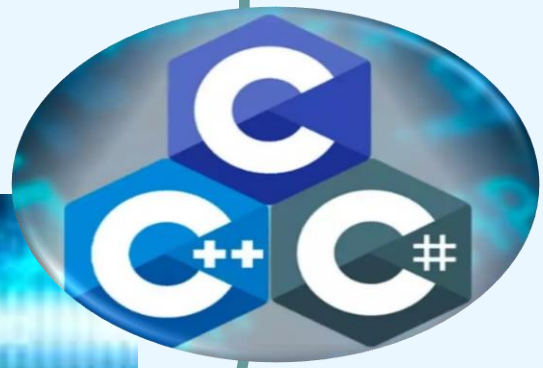
Повторення



```c
#include <stdio.h> // Підключення заголовкових файлів
void exampleFunction(); // Прототипи функцій

int main() { // Головна функція, точка входу
    printf("Hello, world!\n"); // Виведення тексту на екран
    exampleFunction(); // Виклик користувацької функції
    return 0; // Код завершення програми (0 — успішне виконання)
}

// Опис користувацької функції
void exampleFunction() {
        printf("Це виклик додаткової функції.\n");
}
```

Арифметика мови С

Дія в С	Арифметична операція	Алгебраїчний вираз	Вираз на С
Додавання	+	a+7	a+7
Віднімання	-	p-3	p-3
Множення	*	ap	a*p
Ділення	/	a/p або a:p	a/p
Обчислення залишка частки	%	x mod y	x%y

Операції присвоєння

У мові С передбачено декілька операцій присвоєння.

c=c+3; або **c+=3**;

Будь-який оператор виду

змінна*=*змінна операція вираз;

може бути записаний у вигляді

змінна операція* = *вираз;

Операції інкремента та декремента

У мові С передбачена також унарна ***операція інкремента*** **++** та унарна ***операція декремента*** **--**.

x=x+1;	x+=1;	++x;	x++;
y=y-1;	y-=1;	--y;	y--;

Структури управління

Структура вибору використовується для обрання одного з альтернативних напрямків дій.

Структура ***if*** називається структурою з **одиничним** вибором, оскільки в ній вибирається або ігнорується одна дія.

У структурі вибору ***if*** дія, що вказана, виконується тільки тоді, коли умова правдива; у протилежному випадку дія пропускається.

if (умова) {exp1; exp2;}

Структура *if/else*

Структура ***if/else*** називається структурою з подвійним вибором, оскільки в ній вибір відбувається між двома альтернативними діями.

Структура ***if/else*** дає програмістові можливість вказати, що, залежно від того, умова є правдивою або неправдивою, повинні виконуватися різні дії.

```
if (умова)  {
        Блок  1;}
else {
        Блок  2;}
```

Структура із множинним вибором *switch*

Іноді алгоритм включає послідовність прийняття рішень, коли відбувається незалежна перевірка змінної або виразу на рівність кожному зі сталих значень, і залежно від цього виконуються різні дії.

У мові C/C++ для обробки таких ситуацій передбачена структура із множинним вибором ***switch***.

Оператор вибору ***switch*** є дуже зручним засобом заміни множинного використання операторів if.

Загальний формат

```
switch (вираз) {
        case константний вираз1 : блок1;
        case константний вираз2 : блок2;

        …
        default : блок ;
}
```

вираз може бути цілого або символьного типу

Оператори *break* і *continue*

Оператори ***break*** і ***continue*** призначені для зміни ходу виконання дій у програмі.

Виконання оператора ***break*** у структурах ***switch***, ***for***, ***while***, ***do/while*** призведе до негайного виходу зі структури.

Виконання програми продовжується з першого оператора, який записаний після неї.

Виконання оператора ***continue*** у структурах ***for***, ***while***, ***do/while*** призведе до пропуску операторів, що залишилися, у тілі цих структур і виконання наступної ітерації циклу.

Структури повторення

Більшість програм включає повторення, тобто цикли.

Цикл – це група команд, які неодноразово виконуються комп'ютером, поки деяка ***умова продовження*** залишається правдивою.

Оператори, які включені до структури повторень, складають тіло цієї структури.

Тіло структури повторень може бути простим (один оператор) або складеним оператором (блок).

Структури повторення

В мові програмування C/C++ існує **3 структури** повторення

```
for(….) {…;}
```

```
while(…) {…;}
```

```
do {…;} while(…);
```

Структура повторення *for*

Такі повторення іноді називають визначеними повтореннями, оскільки заздалегідь відомо, скільки разів буде виконаний цикл.

Для підрахунку кількості повторень використовується керуюча змінна.

Керуюча змінна змінюється кожний, коли виконується тіло циклу.

Коли значення керуючої змінної показує, що виконана необхідна кількість повторень, цикл завершується, комп'ютер продовжує виконання програми з оператора, який є наступним за структурою повторення.

Загальний формат структури *for*

for (вираз1; вираз2; вираз3) {тіло циклу}

вираз1 ініціює змінну керування циклом,

вираз2 є умовою продовження циклу,

вираз3 вказує, як змінюється змінна керування циклом.

Структура повторення *while*

Структура повторення *while* або повторення, що керуються контрольним значенням

Такі повторення називають невизначеними повтореннями, оскільки заздалегідь невідомо, скільки разів необхідно виконати цикл.

Контрольні значення використовуються для керування кількістю повторень.

Для реалізації структури повторення, що керується контрольним значенням, у мові C/C++ використовуються структури ***while***, ***do/while***.

Формат структури повторення while

у загальному випадку може бути записаний так:

while (вираз) {блок;}

Якщо ***вираз*** має значення, що не дорівнює нулю (тобто **true**), виконуються оператори тіла циклу (блок).

Після цього програма знову повертається до обчислення виразу.

Як тільки значення виразу стало збігатися з 0 (**false**), цикл припиняється, і програма починає працювати з першого оператора після структури ***while***.

Структура повторення *do/while*

Структура повторення ***do/while*** подібна структурі ***while***. У структурі ***while*** умова продовження циклу перевіряється на початку циклу до виконання операторів тіла циклу.

Структура ***do/while*** перевіряє умову продовження циклу **після** виконання тіла циклу, і тому тіло циклу буде **виконано хоча б один раз.**

Після завершення циклу ***do/while*** виконання програми продовжується з оператора, який записаний після речення ***while***.

Формат структури повторення *do/while*

do {блок} while (вираз);

де

блок– послідовність дій (тіло циклу)

вираз - умова продовження циклу

У мові C/C++ **функція** — це набір команд, що виконує певне завдання.

Функції використовують для розбиття коду на логічні блоки, що робить програму **зрозумілою**, **гнучкою** та **легшою для підтримки**.

Більшість функцій мають ***список параметрів***.

Параметри дозволяють функціям обмінюватися інформацією.

Параметри функції – це локальні змінні.

Усі змінні, які об'явлені в тілі функції, є ***локальними змінними*** – вони відомі тільки функції, в якій вони визначені.

Визначення функції

тип_поверненого_значення ім'я_функції (список_параметрів)

{

оголошення

оператори

}

Визначення функції

Як ім'я функції може бути будь-який допустимий ідентифікатор.

Типом результату, який повертає функція, є ***тип поверненого значення***.

Якщо як тип задано ключове слово **void**, це означає, що функція не повертає нічого.

Якщо ***тип поверненого значення*** не вказаний, компілятор вважає, що тип має значення **int**.

Список параметрів - це список об'яв параметрів (відокремлених комами), які отримує функція в момент її виклику.

Якщо функція не отримує значень, ***список параметрів*** позначається ключовим словом **void** або пустими дужками.

Тип кожного параметра повинен бути описаний, за виключенням типу **int**.

Якщо тип не вказаний, вважається, що параметр має тип **int**.

Оголошення та ***оператори*** у середині фігурних дужок складають ***тіло функції***.

Визначення функції

Перед першим викликом функція повинна бути визначена повністю або за допомогою ***прототипу***.

Компілятор використовує прототип функції для перевірки того, що виклик функції має коректний тип поверненого значення, коректне число аргументів, коректний тип аргументів і коректний порядок слідування аргументів.

Якщо функція не повертає результат, управління повертається, як тільки зустрічається права фігурна дужка, що завершує тіло функції,

або при виконанні оператора

return;

Якщо функція повертає результат, оператор

return вираз;// return(вираз);
повертає значення ***виразу***.

```
#include <stdio.h>
void printMessage() {
	printf(" Це функція без параметрів.\n");
}

int main() {
	printMessage(); // Виклик функції
	return 0;
}
```

```c
#include <stdio.h>
// Функція додає два числа і повертає результат
int add(int a, int b) { return a + b; }

int main() {
    int sum = add(5, 7); // Виклик функції
    printf("Сума: %d\n", sum);
    return 0;
}
```

Оголошення масивів

Масив є групою комірок пам'яті, які мають одне і те ж саме ім'я та однаковий тип.

Для використання конкретної комірки або елементу масиву вказується **ім'я масиву та зміщення цієї комірки відносно першої комірки або початку масиву.**

Зміщення вказується після імені масиву у квадратних дужках і називається ***індексом*** масиву.

```
int Array[100], matrix[21];
float Matrix[27];
```

ВИГЛЯД МАСИВУ В ПРОГРАМУВАННІ

оголошення

type name[size];

float A[N];

A[0]	A[1]	A[2]	…	A[k-1]	A[k]	A[k+1]	…	A[N-2]	A[N-1]

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]

Як і змінні інших типів, елементи масиву можуть бути ініційовані значеннями в момент їх оголошення.

Для цього після оголошення масиву поставьте знак рівності "=" і в фігурних дужках напишіть список значень для всіх елементів масиву.

Значення в списку розділяються комами:

```
int AA [12] = {32, 27, 0, 18, 96, 12, 45, 23, 50, 9, 2, 0};


//int AA [] = {32, 27, 0, 18, 96, 12, 45, 23, 50, 9, 2, 0}; n=12;


int A123A [12];
A123A[0]=32; A123A[1]=27; A123A[2]= 0;
A123A[3]= 18; A123A[4]=96; A123A[5]=12;
A123A[6]= 45; A123A[7]=23; A123A[8]=50;
A123A[9]= 9; A123A[10]=2;   A123A[11]=0;
```

ВВЕДЕННЯ/ВИВЕДЕННЯ ОДНОВИМІРНИХ МАСИВІВ

Приклад програми, що дозволяє користувачеві ввести вимірність масиву, а потім його заповнити та вивести на екран

```
#include <stdio.h>

int main () {
    int A[100];
    /* Масив A складається зі
    ста цілих чисел */
    int i,N;
    printf (“Input dimension of the array\n:");
    scanf("%d",&N);
    printf (" Input array N:");
    for (i=0; i <= N-1; i++)// for (i=0; i < N; i++)
        scanf("%d",&A[i]);
     printf (" The rezult array :");
     for (i=0; i <= N-1; i++)
        printf("%3d",A[i]);
     printf("\n");
    return 0;
}
```

Багатовимірні масиви

Оголошення
type name [size1][size2]…[sizeK]

```
int A[3][4];
float B[3][7][2];
double  D[2][2];
char W[4][3];
```

	Стовбець 0	Стовбець 1	Стовбець 2	Стовбець3
Рядок 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Рядок 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Рядок 2	[2][0]	a[2][1]	a[2][2]	a[2][3]

Двовимірний масив
int A[3][4];

	Стовбець 0	Стовбець 1	Стовбець 2	Стовбець3
Рядок 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Рядок 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Рядок 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Багатовимірні масиви, як і одновимірні, можуть бути ініційовані їх оголошенням. Наприклад, оголошення двовимірного масиву ***a*** виглядає наступним чином

int *a*[3][4] = {{1, 2, 7,6}, {3 , 4, 8, 9}, {5, 6, 9,10}};

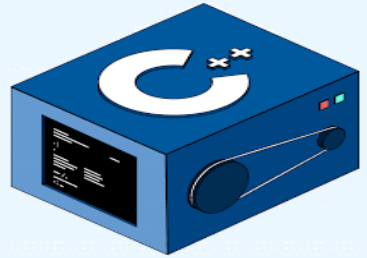
Еквівалентно оголошенню

int *a*[3][4];

та дванадцяті присвоєнням// 3*4=12

***a*[0][0] = 1; *a*[0][1] = 2; *a*[0][2] = 7; *a*[0][3] = 6;**
***a*[1][0] = 3; *a*[1][1] = 4; *a*[1][2] = 8; *a*[1][3] = 9;**
***a*[2][0] = 5; *a*[2][1] = 6; *a*[2][2] = 9; *a*[2][3] =10;**

```
#include <stdio.h>
int main () {
	int a[3][4] = {{1, 2, 7,6}, {3 , 4, 8, 9}, {5, 6, 9,10}};
	int i,j;
	printf (" array by 3x4:\n");
	for (i=0; i <3; i++)
	{
	    for (j=0; j<4;j++)
	        { printf ("%4d",a[i][j]);}
	    printf("\n");
	}
	printf("\n");
	return 0;
}
```



Дякую за увагу!