Міністерство освіти і науки України

Львівський національний університет імені Івана Франка

Факультет електроніки та комп’ютерних технологій

**ЗВІТ**

виконання лабораторної роботи №4

**ТЕМА:** Основні типи змінних. Операції та їх пріоритети.

**МЕТА:** Вивчити поняття: основні типи змінних, вказівники, операції та їх пріоритет.

**СПИСОК ВИКОРИСТАНОГО ПЗ:** онлайн компілятор мови програмування Сі.

**ВИКОНАВ:**

Студент 1-го курсу

Групи ФЕП-14

Іванський Остап Юрійович

**ПЕРЕВІРИВ:**

Асистент

Вдовиченко Віктор Мар’янович

Львів 2024

**ЗМІСТ**

1. Теоретичні відомості.
2. Використання арифметичних та логічних операторів.
3. Програма виведення адреси та значення через вказівник.
4. Програма розв’язку квадратного рівняння.
5. Висновок.

**ХІД РОБОТИ**

**1.Теоретичні відомості.**

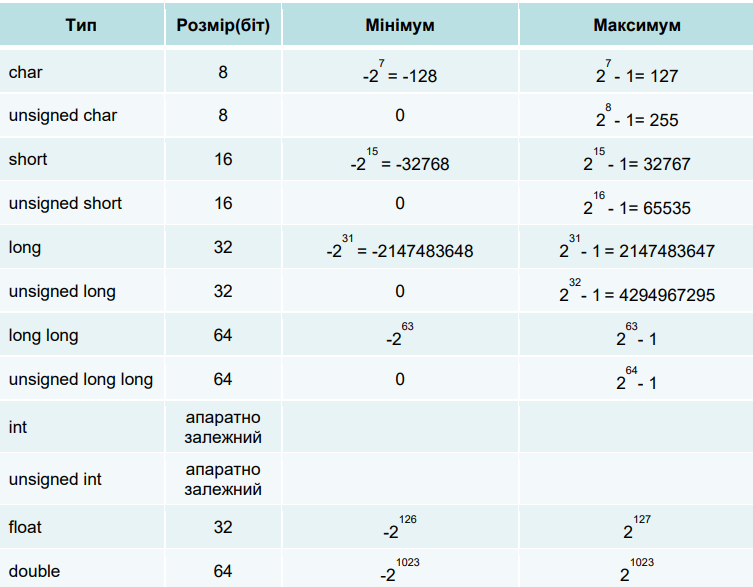
Мова Сі підтримує 5-ть базових типів:

* символьний;
* цілочисельний;
* дійсний з одинарної точністю;
* дійсний з подвійною точністю
* void.

Тип void має три призначення:

* вказівка про неповернення значення функцією (void main());
* вказівка про неотримання параметрів функцією (int main(void));
* створення нетипізованого вказівника (void\*) .

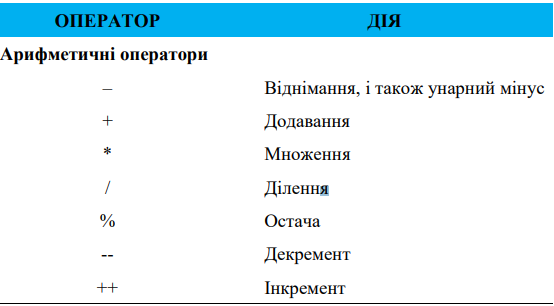
Також існує, ще декілька типів, характеристики котрих наведено в таблиці нижче

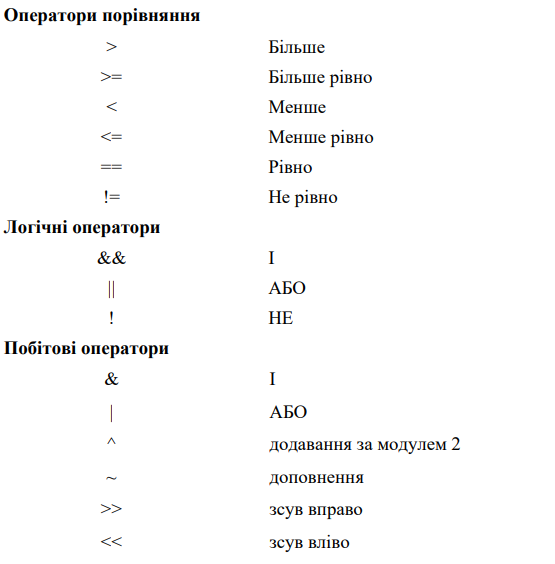


**Оператор** - це символ, який повідомляє компілятору про необхідність виконання деяких математичних або логічних дій.

Є три великі класи операторів: арифметичні, порівняння і логічні, бітові.

Крім цього, Cі має кілька спеціальних операторів.





**Пріоритет** виконання арифметичних операцій наступний:

1. Вищий + (унарний плюс); – (унарний мінус); ++; --; \*; /; %

2. Нижчий +; – (бінарні оператори)

Оператори одного пріоритету, виконуються компілятором зліва направо. Круглі дужки можна використовувати для зміни порядку обчислення. Круглі дужки в С трактуються так само, як і в усіх інших мовах — вони встановлюють найвищий пріоритет операцій.

**Вказівник** – це адреса змінної в пам'яті. Вказівник на змінну – це змінна, спеціально створена для зберігання вказівника на об'єкт певного типу.

Знаючи адресу змінної, можна істотно спростити роботу деяких програм.

Вказівники мають три головних призначення в Сі:

1. Надають швидке звернення до елементів масиву.

2. Дозволяють функції модифікувати передані параметри.

3. Підтримують динамічні структури даних, наприклад списки.

Оператор \*, доповнює &. Це унарний оператор, який повертає значення змінної за вказаною адресою.

Операція взяття адреси:

n = &x; // в змінній n міститься адреса змінної x, тобто n – вказівник

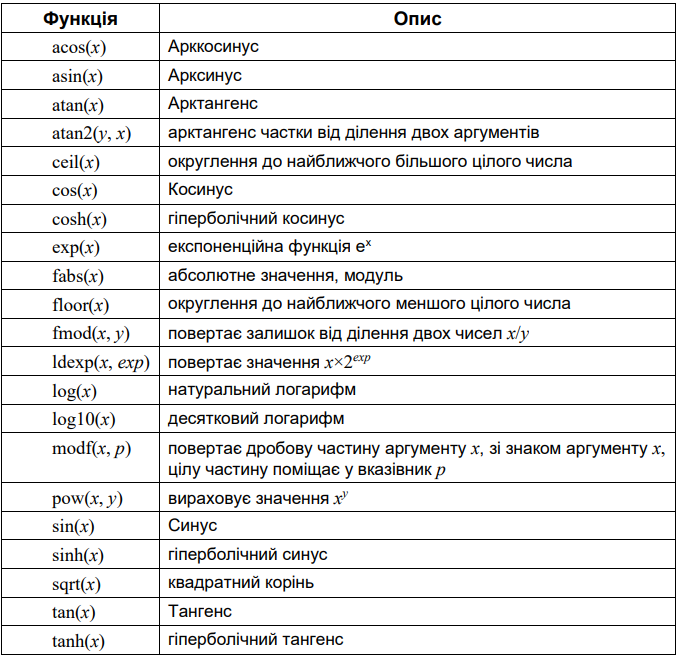
Операція розіменування:

y = \*n; // змінній y присвоюється значення, яке знаходиться за адресою n, –

// тепер змінна y містить значення змінної х

Бібліотека **math**

math.h – заголовний файл стандартної бібліотеки мови програмування Сі, який надає прототипи функцій, розроблених для виконання простих математичних операцій. Більшість функцій використовує числа з плаваючою комою. Усі функції, котрі працюють з кутами, сприймають радіани а не градуси!



**2. Використання арифметичних та логічних операторів.**

Програма яка виводить результат роботи усіх арифметичних та логічних операторів мови Сі.

#include <stdio.h>  
  
void printBinary(int num) {  
 for (int i = 31; i >= 0; i--) {  
 printf("%d", (num >> i) & 1);  
 }  
 printf("\n");  
}  
  
int main(void) {  
 int a = 8;  
 int b = 2;  
  
 // Арифметичні оператори  
 printf("Арифметичні оператори:\n");  
 printf("a + b = %d\n", a + b); // Додавання  
 printf("a - b = %d\n", a - b); // Віднімання  
 printf("a \* b = %d\n", a \* b); // Множення  
 printf("a / b = %d\n", a / b); // Ділення  
 printf("a %% b = %d\n", a % b); // Залишок від ділення  
  
 // Логічні оператори  
 printf("\nЛогічні оператори:\n");  
 printf("(a && b) = %d\n", a && b); // Логічне "І"  
 printf("(a || b) = %d\n", a || b); // Логічне "АБО"  
 printf("!a = %d\n", !a); // Логічне "НЕ"  
  
 // Побітові оператори  
 printf("\nПобітові оператори:\n");  
 printf("a = ");  
 printBinary(a);  
 printf("b = ");  
 printBinary(b);  
  
 printf("a & b = ");  
 printBinary(a & b); // Побітове "І"  
 printf("a | b = ");  
 printBinary(a | b); // Побітове "АБО"  
 printf("a ^ b = ");  
 printBinary(a ^ b); // Побітове "Виключне АБО"  
 printf("~a = ");  
 printBinary(~a); // Побітове "НЕ"  
  
 // Зсуви  
 printf("\nЗсуви:\n");  
 printf("a << 1 = ");  
 printBinary(a << 1); // Лівий зсув  
 printf("a >> 1 = ");  
 printBinary(a >> 1); // Правий зсув  
  
 return 0;  
}

Результат роботи :

Арифметичні оператори:

a + b = 10

a - b = 6

a \* b = 16

a / b = 4

a % b = 0

Логічні оператори:

(a && b) = 1

(a || b) = 1

!a = 0

Побітові оператори:

a = 00000000000000000000000000001000

b = 00000000000000000000000000000010

a & b = 00000000000000000000000000000000

a | b = 00000000000000000000000000001010

a ^ b = 00000000000000000000000000001010

~a = 11111111111111111111111111110111

Зсуви:

a << 1 = 00000000000000000000000000010000

a >> 1 = 00000000000000000000000000000100

Process finished with exit code 0

**3.Програма виведення адреси та значення через вказівник.**

Програма введенням числа у змінну і виведенням її адреси та значення через вказівник.

#include <stdio.h>  
  
int main(void) {  
 int number;  
 int \*ptr;  
  
 // Введення числа  
 printf("Введіть число: \n");  
 scanf("%d", &number);  
  
 // Присвоєння адреси змінної number вказівнику ptr  
 ptr = &number;  
  
 // Виведення значення та адреси через вказівник  
 printf("Введене число: %d\n", \*ptr);  
 printf("Адреса числа: %p\n", (void\*)ptr);  
 return 0;  
}

Результат роботи програми:

Введіть число:

Введене число: 33

Адреса числа: 000000692cfff984

Process finished with exit code 0

**4.Програма розв’язку квадратного рівняння.**

#include <stdio.h>  
#include <math.h>  
  
int main(void) {  
 double a, b, c;  
 double discriminant, root1, root2, realPart, imaginaryPart;  
  
 // Введення коефіцієнтів  
 printf("Введіть коефіцієнти a, b і c: ");  
 scanf("%lf %lf %lf", &a, &b, &c);  
  
 // Перевірка, чи є a ненульовим (якщо a = 0, це не квадратне рівняння)  
 if (a == 0) {  
 printf("Це не квадратне рівняння (a не може дорівнювати 0).\n");  
 return 1;  
 }  
  
 // Обчислення дискримінанта  
 discriminant = b \* b - 4 \* a \* c;  
  
 // Перевірка дискримінанта і обчислення коренів  
 if (discriminant > 0) {  
 // Два різних дійсних корені  
 root1 = (-b + sqrt(discriminant)) / (2 \* a);  
 root2 = (-b - sqrt(discriminant)) / (2 \* a);  
 printf("Корені рівняння є дійсними і різними.\n");  
 printf("Корінь 1 = %.2lf\n", root1);  
 printf("Корінь 2 = %.2lf\n", root2);  
 }  
 else if (discriminant == 0) {  
 // Два однакових дійсних корені  
 root1 = -b / (2 \* a);  
 printf("Корені рівняння є дійсними і однаковими.\n");  
 printf("Корінь = %.2lf\n", root1);  
 }  
 else {  
 // Комплексні корені  
 realPart = -b / (2 \* a);  
 imaginaryPart = sqrt(-discriminant) / (2 \* a);  
 printf("Корені рівняння є комплексними.\n");  
 printf("Корінь 1 = %.2lf + %.2lfi\n", realPart, imaginaryPart);  
 printf("Корінь 2 = %.2lf - %.2lfi\n", realPart, imaginaryPart);  
 }  
  
 return 0;  
}

Результат:

Введіть коефіцієнти a, b і c: Корені рівняння є дійсними і різними.

10

55

21

Корінь 1 = -0.41

Корінь 2 = -5.09

Process finished with exit code 0

**5.Висновок**

Під час роботи над завданням вивчив поняття: основні типи змінних, вказівники, операції та їх пріоритет. Також виконав завдання по створенню програм використавши дані знання.