1. Що таке масив?

Масив - це структура даних, що складається з елементів одного типу. Всі елементи масиву займають суцільний блок пам'яті

1. Що таке оператор індексування?

Елемент масиву можна отримати через його індекс (номер елементу). Нумерація елементів починається з 0

1. Яким є стартовий індекс масиву?
2. Чи можна використовувати змінні для визначення довжини масиву?

Тільки константи

1. Як змінити розмір масиву після створення?
2. Як додати новий елемент в кінець масиву?
3. Як здійснюється передача масивів у функції?

 Ім'я масиву без квадратних дужок є адресою першого елемента масиву. Копія цієї адреси присвоюється відповідному параметру. Функція може використовувати цю адресу для доступу до елементів масиву. Зміни, внесені в елементи, залишаються після завершення функції.

1. Як отримати довжину масиву?

sizeof(arr) / sizeof(arr[0])

1. Як скопіювати значення елементів з одного масиву в інший?
2. Як визначити кількість стовпців в двовимірному масиві?
3. Як ініціалізувати двовимірний масив?
4. Чи можна створити двовимірний масив з різною довжиною рядків?
5. Що таке вказівник?

Вказівник є змінною, яка містить адресу пам'яті. Для визначення вказівника використовують зірочкою, яка передує імені змінної.

1. Як отримати адресу змінної?

Int address = &var

1. Як здійснюється розіменування?

Щоб отримати доступ до даних, що зберігаються за адресою, можна використовувати так зване розіменування (dereferencing). Для розіменування вказівника оператор розіменування (\*) встановлюється перед ім'ям вказівника. Наприклад:

1. Які є варіанти використання вказівників?

Існує особливий тип вказівників: void\*. Можна призначити вказівники будь-яких типів до void\*. Змінні типу void\* використовуються для зберігання вказівників різних типів. У наступному прикладі, масив void\* містить адреси різних змінних:

int i = 0;

double d = 1.5;

char c = 'A';

void \*p[4] = {&i, &d, &c}; // p[3] = 0

double \*pd = &d;

p[3] = pd;

Аргументи функції можуть бути оголошені як вказівники. Це дозволяє надсилати адреси заданих змінних в функцію і змінювати ці змінні, використовуючи їх адреси. Припустимо, що нам потрібно поміняти місцями значення двох цілих змінних. Раніше розглядався приклад функції з використанням посилань. Функція може також використовувати вказівники як параметри:

void swap(int \*p1, int \*p2)

{

int x = \*p1;

\*p1 = \*p2;

\*p2 = x;

}

Вказівники можна використовувати для опису параметра функції типу масиву. Вихідний масив може бути змінений всередині функції:

void modifyStartingElement(int \*p)

{

p[0] = 0;

}

void main()

{

int a[] = {1, 2, 3};

modifyStartingElement(a);

for (int i = 0; i < 3; i++)

{

cout << a[i] << ' '; // 0 2 3

}

}

1. У чому різниця між постійним указівником і вказівником на постійній об'єкт?

Константний вказівник (ключове слово const стоїть після \*) - це вказівник, який не може бути змінений. Проте, значення, на яке вказує вказівник може бути змінене. Константні вказівники обов'язково повинні бути ініціалізовані:

int i = 1;

int \* const cp = &i;

cout << \*cp; // 1

int j = 2;

cp = &j; // Помилка!

Можна створювати вказівники на константний об'єкт: значення, на яке вказує вказівник не може бути змінено, але сам вказівник може бути переміщений на іншу змінну або константу.

int k = 4;

const int \*pk = &k; // Вказівник на константу

k = 5;

cout << \*pk; // 5

\*pk = 6; // Помилка!

1. Як отримати адресу масиву?

Cout << arr;

1. Який є взаємозв'язок між масивами і вказівниками?

Имя массива это указатель на первый элемент

1. Що таке адресна арифметика?

До вказівників можна застосувати деякі операції. Арифметика вказівників обмежується додаванням, відніманням і порівняння. Під час виконання арифметичних операцій з вказівниками передбачається, що вказівники вказують на масив об'єктів. Додавши ціле значення до вказівника, ми переміщуємо його на відповідне число об'єктів у масиві. Якщо, наприклад тип має розмір 10 байт, а потім ми додали ціле число 5, вказівник переміщується на 50 байт в пам'яті. Можна застосувати інкремент і декремент, а також операції складеного присвоювання для неконстантних вказівників.

1. Які області комп'ютерної пам'яті використовують для розташування змінних?

Глобальні змінні знаходяться в глобальному просторі імен, для якого виділяється спеціальний сегмент пам'яті.

Регістри утворюють особливу область пам'яті, вбудовану в центральний процесор.

Стек (stack, стек викликів) це спеціальна область пам'яті, виділена для зберігання даних окремих функцій. Ознакою стеків як структур даних є принцип last-in, first-out (LIFO, першим зайшов, останнім вийшов).

Решта пам'яті, розподіленої для програми, - це так звана динамічна пам'ять (free store).

1. Що таке динамічна пам'ять?

Програмісти можуть використовувати динамічну пам'ять для контрольованого виділення пам'яті й звільнення змінних з пам'яті. Оператор new використовують для розташування змінних у динамічній пам'яті. Операція new повертає вказівник на об'єкт розташований в динамічній пам’яті.

int \*p = new int;

\*p = 65;

1. Як розташувати змінну в динамічній пам'яті?
2. Чому необхідно видаляти непотрібні змінні з динамічній пам'яті?

Після звільнення пам'яті, пам'ять, що раніше була виділення для змінної \*р може бути використана для розташування інших змінних.

1. Як розташувати двовимірний масив у динамічній пам'яті?

// Створення масиву:

int \*\*a = new int\* [m];

for (i = 0; i < m; i++)

{

a[i] = new int [n];

}