Лабораторна робота №21

Покажчики

Мета: набути навичок роботи з покажчиками в одновимірному масиві.

Література

Войтенко В. В., Морозов А. В. C\C++ Практика програмування. Навчально методичний посібник - Житомир: ЖДТУ, 2003. – 324 с.

Зміст роботи

Завдання 1. Написати програму з використанням покажчиків.

1) Оголосити покажчик **p** на комірку пам'яті типу int;
2) Оголосити змінні **x, y** і масив **m**, змінні ініційовані;
3) Покажчику **p** присвоїти адресу змінної **y**.
4) Вивести на екран значення змінної **y** через покажчик;
5) Чому буде дорівнювати **x**, якщо провести операцію **x = *p?**
6) Змінити величину параметра **y** на **7;**
7) Чому буде дорівнювати **p**?
8) Чому буде дорівнювати **y**, якщо провести операцію ***p+=5?**

Завдання 2. Дано масив. Скласти програму де необхідно:

1) Визначити розмір масиву в байтах.
2) Визначити кількість елементів масиву.
3) Вивести на екран адреси першого і останнього елементів масиву.
4) Здійснити переписування масиву у зворотному порядку.

Завдання 3. *Індивідуальні завдання.*

Сформувати одновимірний масив цілих чисел, використовуючи датчик випадкових чисел.

1	1. Видалити елемент з номером K. 2. Додати після кожного парного елемента масиву елемент із значенням 0.
2	1. Видалити перший елемент що дорівнює 0. 2. Додати після кожного парного елемента масиву елемент із значенням mas[i-1]+2.
3	1. Видалити всі елементи що дорівнюють 0. 2. Додати після першого парного елемента масиву елемент із значенням mas[i-1]+2.
4	1. Видалити елементи, індекси яких кратні 3. 2. Додати після кожного негативного елемента масиву елемент із

	значенням mas[i-1]+1
5	1. Видалити із масиву всі елементи які кратні 7. 2. Додати після кожного непарного елемента масиву елемент із значенням 0
6	1. Видалити елемент із заданим номером. 2. Додати після першого парного елемента масиву елемент із значенням mas[i-1]+2.
7	1. Видалити останній елемент що дорівнює 0. 2. Додати після елемента масиву із заданим індексом елемент із значенням 100.
8	1. Видалити всі елементи із заданим значенням. 2. Додати перед кожним парним елементом масиву елемент із значенням 0.
9	1. Видалити перший елемент із заданим значенням. 2. Зсунути масив циклічно на К елементів вправо.
10	1. Видалити 5 перших елементи масиву. 2. Додати в кінець масиву 3 нові елементи.
11	1. Видалити 5 останніх елементів масиву. 2. Додати на початок масиву 3 елементи із значенням mas[i+1]+2.
12	1. Поміняти місцями мінімальний і максимальний елементи масиву. 2. Видалити з масиву всі елементи, що перевищують середнє значення більш, ніж на 10%.
13	1. Видалити з масиву всі елементи, які співпадають із його мінімальним значенням. 2. Додати на початок масиву три елементи із значенням середнього арифметичного масиву.
14	1. Перевернути масив і, якщо число елементів масиву непарне, видалити його середній елемент. 2. Додати на початок масиву 3 елементи із значенням mas[i+10]-2.
15	1. Видалити елемент з номером К. 2. Додати після кожного парного елемента масиву елемент із значенням 0.

Методичні рекомендації

Покажчик – це адреса деякого об’єкту, через яку можна звертатися до цього об’єкту. Операція & дає адресу змінної і використовувати її можна тільки до змінних і елементів масиву.

Приклад:

```
int x,*y; y=&x; //присвоєння адреси змінної x змінній-покажчику y
```

Операція * сприймає свій операнд як адресу деякого об’єкту і використовує цю адресу для вибірки вмісту.

Приклад:

```
int x, *y, z; y=&x; z=*y; //отримання значення за адресою покажчика
```

Покажчики можна використовувати як операнди у арифметичних операціях. Арифметичні операції застосовуються тільки до покажчиків одного типу.

- Інкремент збільшує значення покажчика на величину sizeof(тип).

```
char* pc;

int* pi;

double* pd;

...

pc++; //значення збільшується на 1

pi++; // значення збільшується на 4

pd++; // значення збільшується на 8
```

- Декремент зменшує значення покажчика на величину sizeof(тип).
- Різниця двох вказівників - це різниця їх значень, поділена на розмір типу в байтах. Підсумовування двох покажчиків не допускається.
- Можна підсумувати покажчик і константу

Конструкція y+n (y – покажчик, n – ціле число) задає адресу n-го об’єкту, на який вказує y. Це справедливо для будь-яких об’єктів (int, char, float і т.п.).

Будь-який покажчик можна перевірити на рівність (==) або нерівність (!=) зі спеціальним значенням NULL.

```
int *y; y=NULL; int x=5; if(y==NULL) y=*x;
```

Після створення покажчика в ньому знаходиться довільне значення, тобто він посилається на будь-який фрагмент пам’яті. Якщо по тексту програми покажчик перевіряється на NULL, то після створення покажчика його необхідно приводити до значення NULL.

Зв’язок масивів і покажчиків

Якщо задати: *int mas[100], *p, a;* то:

1) для масиву відводиться пам’ять в адресному просторі під 100 елементів типу int;

2) пам’ять відводиться під покажчик-константу з ім’ям mas, значенням покажчика є адреса масиву;

3) пам’ять відводиться під покажчик-змінну з ім’ям p.

Операція ініціалізації покажчика може здійснюватися тільки операцією присвоєння адреси деякої змінної:

p=&a;

p=&mas[0]; або p=mas; або присвоєнням *p=NULL*.

Допустимо p=0, але не рекомендується.

Помилкою є:

```
a=10;
p=a; // де p – покажчик. Присвоєння неможливе, так як типи int* і int.
p=10; // присвоєння неможливе, так як тип int* и const int.
```

Покажчику неможна присвоювати цілі значення, але можна додавати і віднімати покажчик і цілі числа.

Наприклад:

- *p+=10; збільшує на 10 вміст комірки пам’яті, на яку посилається p, еквівалентно mas[0]=mas[0]+10;
- якщо p=mas; то p+=10; еквівалентно p=p+10 і еквівалентно присвоєнню p=&mas[10];

Якщо 2 покажчика посилаються на елементи одного і того ж масиву, то допускаються операції відношення над ними: ==; !=; <, >, і т. п.

Покажчик можна присвоїти іншому покажчику:

```
int x;
int *p1, *p2;
p1 = &x;
p2 = p1;
printf(" %p", p2); /* Виводить адресу змінної x, не її значення */
```

У прикладі на змінну x посилаються обидва покажчика p1 і p2.

Динамічне виділення пам’яті під масиви

Змінні у програмах повинні розміщатися в одному з трьох місць: в області даних програми, в області стеку, в області вільної пам’яті (купи).

Кожній змінній у програмі може відводитися пам’ять або статично (в момент завантаження), або динамічно (у процесі виконання програми).

До цих пір всі масиви оголошувались статично, а отже, зберігали значення своїх елементів в області даних. Якщо кількість елементів невелика, таке розміщення виправдано. Але досить часто виникають випадки, коли необхідно мати великі масиви даних або розмір масиву заздалегідь не може бути визначений. Тут на допомогу приходить можливість використання динамічної пам'яті.

Для того, щоб у пам'яті можна було розмістити будь-який динамічний об'єкт, для нього необхідно попередньо виділити відповідне місце. По завершенні роботи з об'єктом виділену пам'ять необхідно звільнити.

Виділення пам'яті можна здійснювати за допомогою функцій ***malloc(), calloc()***, ***free()***. Ці функції описані у заголовочному файлі <*stdlib.h*> або <*malloc.h*>

```
void* malloc(size_t size);
void* calloc(size_t n, size_t size);
void* realloc(void* ptr, size_t size);
```

Функція *malloc(size)* виділяє *size* байтів з купи. У випадку успішного виділення пам'яті покажчик встановлюється на виділений блок пам'яті. При невдалому виділенні пам'яті функція повертає *NULL*.

Функція *calloc(num, size)*, окрім виділення області пам'яті під масив об'єктів, ще здійснює ініціалізацію елементів масиву нульовими значеннями. *num* вказує, скільки елементів буде зберігатися у масиві, а *size* – розмір кожного елемента у байтах.

Динамічне виділення пам'яті для одновимірного масиву цілих чисел розміром *SIZE* і заповнення елементів масиву:

```
int* p1 = NULL;
int* p2 = NULL;
p1= (int*) malloc(sizeof(int) * SIZE);
//p1= (int*) calloc(SIZE, sizeof(int));
p2 = p1;
printf("\n---------------p1-----------------\n");
for (int i = 0; i < SIZE; i++)
{
      p1[i] = i;
      printf("%d ", p1[i]);
}
printf("\n---------------p2-----------------\n");
int  SIZE_N=SIZE +5;
p2 = (int*) realloc(p1, SIZE_N*sizeof(int));
for (int i = SIZE; i < SIZE_N; i++)
      p2[i] = i;
```

Функція *free(p)* звільняє пам'ять, на яку вказує покажчик.

Контрольні питання:

1. В чому різниця між статичними та динамічними масивами?
2. Коли і звідки виділяється пам’ять для статичних і динамічних масивів?
3. Як оголосити одновимірний динамічний масив?
4. Як визначити розмір одновимірного масиву?
5. Наведіть фрагмент коду для заповнення масиву цілими (дійсними) випадковими числами.

Завдання на самостійну роботу:

Опрацювати лекції №14, 15, 16

1. Створити одновимірний масив. Поміняйте місцями елементи з парними і непарними індексами.
2. Створити два масиви x[a], y[b]. Створіть нові масиви, які будуть вміщати:
   a. елементи обох попередніх масивів
   b. їх спільні елементи