Лабораторна робота №12
Робота з одномірними масивами

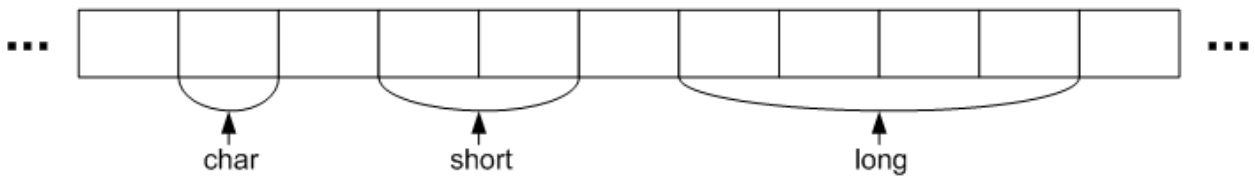
Мета роботи: отримання практичних навичок роботи з одномірними масивами.

Зміст:

Короткі теоретичні відомості

Вказівник

Пам'ять комп'ютера можна уявити як пронумеровану послідовність комірок, з якими можна працювати окремо або пов'язаними областями. Для будь-якої обчислювальної машини буде вірним таке твердження: один байт може зберігати значення типу **`char`**, двобайтові комірки можуть розглядатися як ціле типу **`short`**, а чотирьохбайтові – як цілі типу **`long`**.



Довідка. *Існують архітектури, на яких не будь-які довільні сусідні комірки можуть зберігати ціле типу* ***short****, а лише ті, що починаються з парної адреси – так зване вирівнювання. Так само може застосовуватися і до більших типів.*

Вказівник – це адреса пам'яті, виділеної для розміщення ідентифікатора (як ідентифікатор може виступати ім'я змінної, масиву і т.д.). У тому випадку, якщо змінна оголошена як вказівник, то вона містить адресу пам'яті, за якою можна знаходити скалярні величини будь якого типу.

При оголошенні змінної типу вказівник, необхідно визначити

- тип об'єкта даних, адресу якого буде містити змінна,
- ім'я вказівника з попередньою зірочкою (або групою зірочок).

Формат оголошення вказівника:

```
специфікатор-типу [ модифікатор ] * описувач
```

Специфікатор-типу задає тип об'єкту та може бути будь-якого основного типу, типу структури, тощо. Задаючи замість *специфікатору-типу* ключове слово **void**, можна своєрідним чином відстрочити специфікацію типу, на який посилається вказівник. Змінна, яка оголошується як вказівник на тип **void**, може бути використана для посилання на об'єкт будь-якого типу. Однак для того, щоб можна було виконати арифметичні і логічні операції над вказівниками або над об'єктами, на які вони вказують, необхідно при виконанні кожної операції явно визначити тип об'єктів. Такі визначення типів може бути виконано за допомогою операції приведення типів.

В якості *модифікаторів* при оголошенні вказівника можуть виступати ключові слова **const, near, far, huge**. Ключове слово **const** вказує, що вказівник не може бути змінений в програмі. Розмір змінної оголошеної як вказівник, залежить від архітектури комп'ютера і від моделі пам'яті що використовується на машині, для якого буде компілюватися програма.

Для модифікації розміру вказівника можна використовувати ключові слова **`near, far, huge`** (залежить від платформи).

Приклади оголошення вказівників:

```
int           *p_i;       // вказівник на ціле типу int
float         *p_f;       // вказівник на дробове типу float
unsigned char * const w = &obj; // змінна w оголошена як константний
                                // вказівник на змінні типа char unsigned.
```

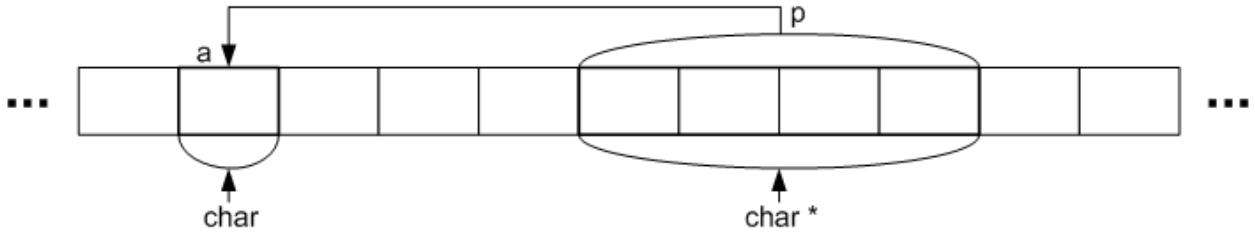
Оскільки вказівники містять адреси, то при використанні однієї і тієї ж моделі розподілу пам'яті, змінні які є вказівниками займають область пам'яті однакового розміру, незалежно від типу даних, на які вони вказують.

Операція взяття адреси об'єкта

Унарний оператор & видає адресу об'єкта.

Наприклад:

```
chara    *p;
char     a;
p = &a;     // p присвоюється значення адреси змінної a
```



Унарний оператор & може бути застосований тільки до об'єктів, розташованих в пам'яті комп'ютера (змінних, елементів масиву). Його операндом не може бути вираз або константа.

Операція непрямого доступу

При роботі з пам'яттю також використовується оператор непрямого доступу (*). Застосований до вказівника, він дозволяє отримати доступ до об'єкта, адреса якого записана у вказівнику. Оператор непрямого доступу застосовувати ТІЛЬКИ до вказівників. Результатом цієї операції буде той об'єкт, на який вказує вказівник. Оператор непрямого доступу не можна застосовуватися до вказівників на тип **`void`**.

Наприклад:

```
char       a;
char   *p = &a;
a = 5;     // записати в змінну a число 5
*p =5;     // записати 5 в комірку, адреса якох зберігається в p тобто
           // в p зараз зберігається адреса змінної a, то ефект від
           // останнього виразу буде аналогічний попередньому виразу
```

Унарні оператори * і & мають вищий пріоритет, ніж арифметичні оператори.

У деякий момент програми вказівник може не містити ніякої реальної адреси. У таких випадках, для того щоб уникнути непорозумінь, використовують ініціалізацію вказівника за допомогою спеціальної константи **`NULL`**, яка визначена в одному з файлів із заголовками стандартної бібліотеки (як правило у файлі *stdlib.h, stdio.h*).

Наприклад:

```
char    *p = NULL;
```

Вирази з вказівниками

З вказівниками можна виконувати деякі дії:

1. До вказівників можна застосовувати операцію присвоювання. Вказівнику на **`void`** можна присвоювати вказівник будь-якого типу. Однак при зворотному присвоюванні необхідно використати явне перетворення типів.
2. Над вказівниками можна виконувати арифметичні операції:
    a. Додавання (вказівник + ціле) та віднімання (вказівник - ціле).
    b. Віднімання - (вказівник - вказівник). Якщо *p1* та *p2* – вказівники на елементи одного і того ж масиву, то операція *p1-p2* дає такий же результат, що і віднімання індексів відповідних елементів масивів.
    c. Інкремент чи декремент (збільшення або зменшення): ++, --. При використанні інкремента, вказівник після кожного збільшення буде вказувати на наступний елемент базового типу. При використанні декремента, вказівник після кожного зменшення буде вказувати на попередній елемент базового типу.

    Інші арифметичні операції над вказівниками заборонені, наприклад, не можна додавати два вказівники, помножити вказівник на число та інше.
3. Вказівники можна порівнювати. Можна застосовувати всі 6 операцій порівняння:>,> =, <, <=, =, ==,! =. Зазвичай порівняння вказівників використовується, коли два або більше вказівників вказують на один об'єкт, наприклад масив.

Масиви

Масиви використовуються тоді, коли потрібно обробляти деяку кількість однотипних об'єктів (наприклад, список успішності студентів).

Масив – це група елементів однакового типу (**`double`**, **`float`**, **`int`** та т.п.). Під час оголошення масиву компілятор повинен отримати інформацію про тип елементів масиву і їх кількість. Оголошення масиву має два формати:

```
специфікатор-типу описувач [константний-вираз];
специфікатор-типу описувач [ ];
```

Описувач – це ідентифікатор (ім'я) масиву.

Специфікатор-типу задає тип елементів масиву що оголошується. Елементами масиву не можуть бути функції і елементи типу **void**.

Константний-вираз у квадратних дужках задає кількість елементів масиву. Константне-вираз при оголошенні масиву може бути опущено в наступних випадках:

- при оголошенні масив що ініціалізується,
- масив оголошений як формальний параметр функції,
- масив оголошений як посилання на масив, явно визначений у іншому файлі.

Приклад: оголосимо масив з 10 елементів типу **int**.

```
int a[10];
```

Виділяється блок з 10 послідовних об'єктів з іменами *a [0], a [1], ..., a [9]*. Зауважте, що оскільки для зберігання числа типу **int** потрібно 4 байти, то реально для збереження цього масиву потрібно 40 байт.

У мові *C* між вказівниками і масивами існує тісний зв'язок. Наприклад, коли оголошується масив у вигляді *int array[25]*, то цим визначається не тільки виділення пам'яті для двадцяти п'яти елементів масиву, але й створення вказівника з ім'ям *array*, значення якого дорівнює адресі першого за рахунком (нульового за індексом) елемента масиву, причому доступ до елементів масиву здійснюється через вказівник з ім'ям *array*. З точки зору синтаксису мови вказівник *array* є константою, значення якої можна використовувати у виразах, але змінити це значення не можна.

Для доступу до елементів масиву існує два різні способи. Перший спосіб пов'язаний з використанням звичайних індексних виразів у квадратних дужках, наприклад,

```
array[16]=3;
```

або

```
array[i+2]=7;
```

При такому способі доступу використовують дві величини, причому друга величина записується у квадратні дужки. Одна із цих величин має бути вказівником, а друга - виразом цілого типу. Послідовність запису цих виразів може бути будь-якою. Тому записи *array*[16] та 16[*array*] будуть еквівалентними для компілятора та позначають елемент масиву з номером шістнадцять.

Інший спосіб доступу до елементів масиву пов'язаний з використанням адресних виразів та операції непрямого доступу у формі

```
*(array+16)=3;
```

або

```
*(array+i+2)=7;
```

При такому способі доступу адресний вираз доступу до шістнадцятого елемента масиву теж може бути записано різними способами **(array+16)* або **(16+array)*.

При аналізі програми компілятором, перший спосіб приводиться до другого, тобто індексний вираз перетворюється у адресний. Для наведених прикладів *array[16]* та *16[array]* перетворюються в **(array+16)*.

У мові *C* не проводиться перевірка меж масивів: таким чином, ніщо не зупинить вас при виході за межі масиву. Якщо вихід за межі масиву відбувається під час виконання оператора присвоювання, то інші змінні чи текст програми можуть бути спотворені. Таким чином, на програміста лягає відповідальність за правильність роботи з масивами!

Елементи масиву, зазвичай, перебираються за допомогою циклів.

Ініціалізація масиву

Терміном «ініціалізація» позначають можливість задати початкові значення елементів масиву без програмування відповідних дій. Наприклад, не вдаючись до програмних засобів типу присвоєння значень в циклі або зчитування даних із зовнішнього джерела (файл, клавіатура, блок даних).

В *C* одночасно з оголошенням масиву можна задати початкові значення всіх елементів масиву або лише кількох перших його компонент.

Наприклад:

```
int a[3]={1,2,3};
int d[10]={1,2,3};
```

rand — генерація псевдовипадкових чисел

```
int rand( void );
```

Функція *rand* повертає псевдовипадкове ціле число в діапазоні від 0 до **RAND_MAX** (32767). Використовуйте функцію *srand* для кращої ініціалізації генератора перед викликом *rand*.

Для функціонування *rand* потрібно включити в текст програми файл із заголовками *<stdlib.h>*..

Приклад

```
// crt_rand.c
// This program seeds the random-number generator
// with the time, then displays 10 random integers.
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

int main( void )
{
    int i;
    // Seed the random-number generator with current time so that
    // the numbers will be different every time we run.
    srand( (unsigned)time( NULL ) );

    // Display 10 numbers.
    for( i = 0;   i < 10;i++ )
        printf( "  %6d\n", rand() );
    printf("\n");

    // Usually, you will want to generate a number in a specific range,
    // such as 0 to 100, like this:
    {
        int RANGE_MIN = 0;
        int RANGE_MAX = 100;
        for (i = 0;    i < 10; i++ )
        {
            int rand100 = (((double) rand() /
                (double) RAND_MAX) * RANGE_MAX + RANGE_MIN);
            printf( "  %6d\n", rand100);
        }
    }
}
```

Робоче завдання

Написати програму, в якій:

1. Задати масив цілих чисел довжиною згідно варіанту.
2. Елементи масиву задати випадковим чином в межах заданих діапазонів*.
3. Вивести згенерований масив на екран.
4. Виконати обробку масиву відповідно до варіанту та вивести оброблений масив на екран.

* Для цілей налагодження можна користуватися наперед проініціалізованим масивом, але захищати програми потрібно із генерацією випадкового масиву

Номер варіанта	Довжина масиву	Діапазон значень елементів масиву (цілі величини)	Завдання
1	20	-3…10	Для кожного позитивного елементу масиву знайти, чи є в масиві елементи, рівні йому за модулем, але протилежні за знаком. Якщо є, то змінити знак негативного елементу. Вивести елементи масиву на екран.
2	22	-5…10	Визначити, яке абсолютне значення найчастіше зустрічається в масиві і вивести його на екран.
3	25	-10…20	Знайти максимальний та мінімальний елементи масиву і поміняти їх місцями. Вивести елементи масиву на екран. ПРИМІТКА. Якщо в масиві є кілька елементів, які відповідають мінімальному або максимального значення, то міняються місцями перші з них.

Номер варіанта	Довжина масиву	Діапазон значень елементів масиву (цілі величини)	Завдання
4	17	-10…20	Підрахувати кількість пар сусідніх елементів, які мають протилежні знаки, і вивести ці пари на екран (із зазначенням індексу кожного виведеного елемента). ПРИМІТКА. Вважати нуль позитивним числом.
5	20	-11…20	Знайти суму чисел, що знаходяться між максимальним і мінімальним елементами масиву (в суму включити ці елементи). Вивести суму на екран. ПРИМІТКА. Якщо в масиві є кілька елементів, які відповідають мінімальному або максимального значення, то використовувати перші з них.
6	23	-5…10	Елемент вектора називається локальним максимумом, якщо він строго більше двох своїх сусідів. Знайти всі локальні максимуми в межах масиву і вивести на екран номери локальних максимумів, індекси масиву їм відповідні, а також значення відповідних елементів масиву.
7	25	-5…15	Вивести на екран індекси всіх парних елементів масиву і самі значення цих елементів. ПРИМІТКА. Вважати нуль парним числом.

Номер варіанта	Довжина масиву	Діапазон значень елементів масиву (цілі величини)	Завдання
8	24	-8…15	Поміняти місцями 1-й позитивний елемент з останнім позитивним елементом, 2-й - передостаннім і т.д. Вивести елементи масиву на екран. ПРИМІТКА. Вважати нуль позитивним числом.
9	25	-10…25	Вивести на екран індекси всіх непарних елементів масиву і самі значення цих елементів. ПРИМІТКА. Вважати нуль парним числом.
10	22	-5…15	Для кожного позитивного елементу масиву знайти чи є в масиві елементи, рівні йому за модулем, але протилежні за знаком. Якщо є, то змінити знак і позитивних, і негативних елементів на протилежний. Вивести елементи масиву на екран.

Номер варіанта	Довжина масиву	Діапазон значень елементів масиву (цілі величини)	Завдання
11	19	-10…20	Знайти середньоарифметичне чисел, що знаходяться між максимальним і мінімальним елементами масиву (включаючи ці елементи в обчислення середньоарифметичного), і вивести це значення на екран. ПРИМІТКА 1. Якщо в масиві є кілька елементів, які відповідають мінімальному або максимального значення, то виконувати розрахунки для перших з них. ПРИМІТКА 2. Середньоарифметичне розраховувати в дійсному форматі.
12	24	1…30	Вивести на екран елементи та індекси елементів, значення яких лежать в межах хоча б в 1,5 рази більше найменшого елемента масиву, але не більше, ніж 0,5 найбільшого елемента масиву.
13	25	-10…25	Поміняти місцями 1-й негативний елемент з останнім негативним елементом, 2-й - передостаннім і т.д. Вивести елементи масиву на екран. ПРИМІТКА. Вважати нуль позитивним числом.
14	22	-10…20	Вивести на екран елементи та індекси елементів, які менше середнього значення всіх елементів масиву.

Номер варіанта	Довжина масиву	Діапазон значень елементів масиву (цілі величини)	Завдання
15	20	-6…26	Визначити суму елементів масиву, кратних п'яти. Вивести цю суму на екран.
16	17	-10…20	Знайти суму чисел, що знаходяться між максимальним і мінімальним по модулю елементами масиву (в суму включити ці елементи). Вивести отриману суму на екран. ПРИМІТКА 1. Якщо в масиві є кілька елементів, які відповідають мінімальному по модулю або максимальному по модулю значенням, то виконувати розрахунки для перші з них.
17	23	0…18	Вивести на екран значення та індекси елементів, які є степенями 2.
18	25	0…16	Визначити суму елементів масиву, кратних трьом. Вивести отриману суму на екран.
19	24	-10…20	Замінити кожен елемент масиву на середньоарифметичне між ним і його сусідами справа і зліва ($a_n = \frac{a_{n-1} + a_n + a_{n+1}}{3}$). Перший і останній елементи масиву не змінювати. Вивести елементи масиву на екран.
20	25	-15…32	Знайти суму унікальних (зустрічаються тільки один раз в масиві) елементів масиву.

Номер варіанта	Довжина масиву	Діапазон значень елементів масиву (цілі величини)	Завдання
21	20	-7…15	Елемент вектора називається локальним мінімумом, якщо він строго менше двох своїх сусідів. Знайти всі локальні мінімуми в межах масиву і вивести на екран номери локальних мінімумів, індекси масиву їм відповідні, а також значення відповідних елементів масиву.

Контрольні питання

1. Який тип має результат, що повертається після операції взяття адреси?
2. Який оператор треба використовувати, щоб поміняти значення змінної, знаючи її адресу?
3. Які дії припустимі із вказівниками?
4. Чи можна перемножити два вказівники?
5. Що спільного і в чому відмінність між вказівниками і масивами?
6. Які способи звернення до елементів масиву вам відомі?