МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота № 4

з дисципліни
«Алгоритмізація та програмування»

Виконав:
студент групи КН-108
Мокрик Ярослав
Викладач:
Гасько Р. Т.

Львів – 2018 р.

Тема: "Робота з одновимірними масивами"

Мета: Одержання навичок обробки одновимірних масивів.

• Короткі теоретичні відомості

1.1. Визначення масиву

Визначення масиву містить тип елементів, ім'я масиву й кількість елементів у масиві.

int mas[10];

0	1	2	3	4	5	6	7	8	9

Тобто індекси елементів у масиві mas можуть змінюватися від 0 до 9, усього в масиві 10 елементів.

1.2. Ініціалізація масиву

Ініціалізація масивів можлива при їхньому визначенні:

double d[] = {1, 2, 3, 4, 5};

Довжина масиву обчислюється компілятором за кількістю значень перерахованих у фігурних дужках.

1.3. Вказівники

Кожна змінна в програмі це об'єкт, що має ім'я й значення. Через ім’я можна звернутися до змінної й отримати її значення. Оператор присвоювання (=) виконує зворотнью дію: імені змінної ставиться у відповідність значення.

a=10;

Вираз &a дозволяє отримати адресу ділянки пам'яті, виділеного змінній a. Операція & застосовна тільки до об'єктів, які мають ім'я й розташовані у пам'яті.

Маючи можливість визначити адресу змінної за допомогою &, потрібно мати можливість працювати із цією адресою: зберігати її, передавати, перетворювати. Для цього вводиться поняття вказівника. Вказівник - це змінна, значенням якої служить адреса об'єкта

конкретного типу. Нульова адреса позначається константою NULL, що визначена в заголовковому файлі stdio.h. Щоб визначити вказівник треба повідомити на об'єкт якого типу посилається цей вказівник.

```
char *z;
int *k,*i;
float *f;
```

* - це операція разіменування. Операндом цієї операції завжди є вказівник. Результат операції - це той об'єкт, який адресує вказівник_операнд.

```
*z=’$ ‘;
```

```
*k=*i=0;
```

Приклад:

```
int e, c, b, *m;
. . . . . .
m = &e ;
*m = c + b ;
```

Операції над вказівниками.

- присвоювання (=);
- отримання значення об'єкта, на який посилається вказівник (*);
- отримання адреси самого вказівника (&).

Приклад:

```
int date = 10;
int *i, *k;;
i = &date;
k = i;
z = NULL;
```

Подібно до будь-яких змінних змінна типу вказівник має ім'я, аресу в пам'яті й значення.

За допомогою унарних операцій ++ і -- числові значення змінних типу вказівник змінюються по різному, залежно від типу даних, з яким пов'язані ці змінні.

Приклад:

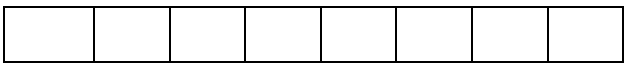
```
char *z;
int *k,*i;
float *f;
. . . . . .
z++; // значення змінюється на 1
i++; // значення змінюється на 2
f++; // значення змінюється на 4
```

Тобто при зміні вказівника на 1, вказівник переходить до початку наступного (попереднього) поля тієї довжини, що визначається типом об'єкта, адресованого вказівником.

1.4. Вказівники й масиви

Ім'я масиву без індексу є вказівником-константою, тобто адресою першого елемента масиву (a[0]).

a



*a = = a[0] ;

*(a+1) = = a[1];

. . . . . .

*(a+i) = =a[i];

Відповідно до синтаксису в С існують тільки одновимірні масиви, але їхніми елементами, у свою чергу, теж можуть бути масиви.

int a[5][5];

Для двовимірного масиву:

a[m][n] = = *(a[m]+n) = = *(*(a+m)+n);

Варіант завдання

19.

- Реалізувати з використанням масиву однонаправлене кільце (перегляд можливий зправа наліво, від першого елемента можна перейти до останнього).
- Роздрукувати отриманий масив, починаючи з К-ого елемента і до К+1.
- Додати в кільце перший і останній елементи.
- Знищити з кільця непарні елементи.
- Роздрукувати отриманий масив, починаючи з К-ого елемента й до К+1.

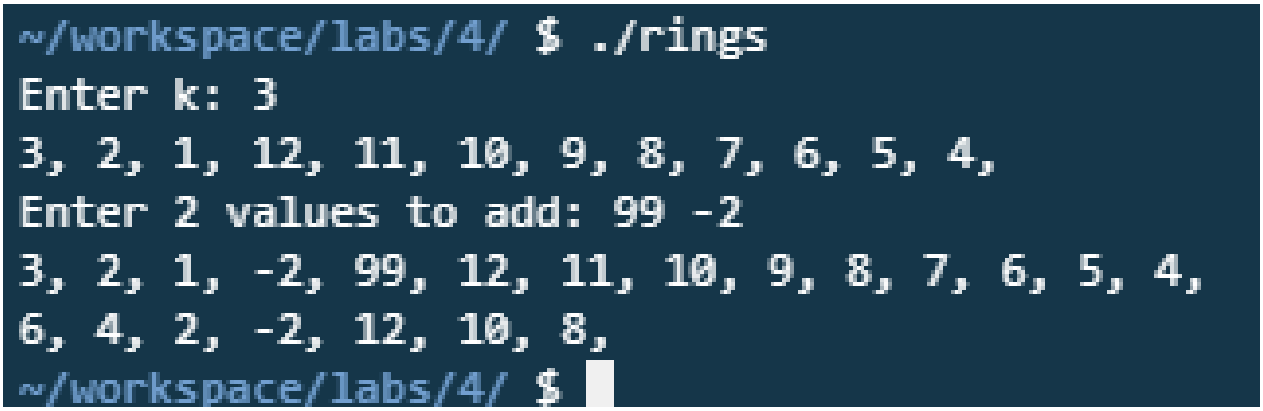
Текст програми

```
#include <stdio.h>
#include <math.h>

void printRing(int *ring, int len, int k) {
    for (int j = 0; j < len; j ++) {
        k--;
        if (k == -1)
            k = len - 1;
        printf("%d, ", ring[k]);
    }
    printf("\n");
}
```

```
int main() {
    int length = 12, k;
    int ring[100] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
    printf("Enter k: ");
    scanf("%d", &k);
    printRing(ring, length, k);
    printf("Enter 2 values to add: ");
    int a, b;
    scanf("%d %d", &a, &b);
    ring[length] = a;
    ring[length + 1] = b;
    length += 2;
    printRing(ring, length, k);
    for (int i = 0; i < length; i ++) {
        ring[i] = ring[i * 2 + 1];
    }
    length = length / 2;
    printRing(ring, length, k);
    return 0;
}
```

Результати



CS50 Тиждень 3
Текст програми

```
/**
 * helpers.c
 *
 * Computer Science 50
 * Problem Set 3
 *
```

```
 * Helper functions for Problem Set 3.
 */

#include <cs50.h>
#include "helpers.h"

bool search(int value, int values[], int n)
{
    int start = 0, end = n-1, middle = (end+start+1)/2;
    while (start <= end) {
        if (values[middle] == value)
            return true;
        if (values[middle] < value)
            start = middle + 1;
        else
            end = middle - 1;
        middle = (end+start+1)/2;
    }
    return false;
}

void sort(int values[], int n)
{
    int temp, counter = 1;
    while (counter > 0) {
        counter = 0;
        for (int i = 0; i < n-1; i ++) {
            if (values[i] > values[i+1]) {
                temp = values[i];
                values[i] = values[i+1];
                values[i+1] = temp;
                counter ++;
            }
        }
    }
    return;
}
```

Результат

ВІРНО

Правильно

:) helpers.c exists.
:) helpers.c compiles.
:) sorts {5,4,3,2,1}
:) sorts {5,3,1,2,4,6}
:) finds 28 in {28,29,30}
:) finds 28 in {27,28,29}
:) finds 28 in {26,27,28}
:) finds 28 in {27,28,29,30}
:) finds 28 in {26,27,28,29}
:) finds 28 in {25,26,27,28}
:) doesn't find 28 in {25,26,27}
:) doesn't find 28 in {25,26,27,29}
:) doesn't find 28 in {29,30,31,32}
:) doesn't find 28 in {29, 30, 31}
:) finds 28 in {30,27,28,26}