


Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ  
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)  
Кафедра экономической математики, информатики и статистики (ЭМИС)

УКАЗАТЕЛИ И ДИНАМИЧЕСКИЕ МАССИВЫ  
Отчёт по лабораторной работе по дисциплине “Введение в  
программирование”

Студент гр. 543-1

 /А.Е. Мухамеджан

“ \_\_\_\_ ” \_\_\_\_\_ 2023

Доцент кафедры ЭМИС

\_\_\_\_\_ Е.А. Шельмина  
оценка

“ \_\_\_\_ ” \_\_\_\_\_ 2023 г.

Томск 2023

## **Лабораторная работа №8**

### **“Указатели и динамические массивы”**

Цель работы: получение навыков работы с указателями и динамическими массивами.

Указатели: Указатель (pointer) – переменная, диапазон значений которой состоит из адресов ячеек памяти или специального значения – нулевого адреса. Если переменная объявлена как указатель, то она содержит адрес памяти, по которому может находиться скалярная величина любого типа. При объявлении переменной типа указатель, необходимо определить тип объекта данных, адрес которых будет содержать переменная, и имя указателя с предшествующей звездочкой.

Нулевой указатель – это указатель, хранящий специальное значение (0 или NULL), используемое для того, чтобы показать, что данная переменная-указатель не ссылается (не указывает) ни на какой объект.

Основные действия над указателями: Для того чтобы присвоить указателю адрес некоторой переменной нужно использовать унарную операцию &. С использованием этой операции мы уже раньше встречались в функции scanf(). Присвоение указателю адреса переменной можно осуществлять как в месте его объявления, так и отдельным оператором.

Динамические массивы: динамические массивы могут быть созданы двумя способами: либо с помощью операции new() из языка C++, либо с помощью функции malloc() из библиотеки языка C, при этом нужно указать тип и количество элементов массива.

Создаётся указатель, в операционной памяти выделяется непрерывная область смежных ячеек памяти для размещения элементов указанного типа, и при этом адрес начальной ячейки записывается в указатель.

Аналогично функция malloc(m) из языка C выделяет непрерывную область памяти длиной m байт.

Доступ к элементам динамического массива осуществляется точно так же как и к статическим, так как массив и указатель одно и то же.

Динамические массивы не обнуляются при создании.

Память, выделенная для динамического массива, после использования должна быть освобождена. Для первого способа (для операции `new()`) с помощью оператора `delete()`, для второго способа (для функции `malloc()`) посредством функции `free()`.

Операции с указателями: на рисунке 1 собраны примеры основных операций и функций, применяемых в языке Си с указателями. В примерах подразумевается, что указатели типа `int*`, но эти операции применимы и для других типов данных.

Пример операции	Описание
<code>ptr = &amp;var;</code>	Присвоить указателю <b>ptr</b> адрес переменной <b>var</b>
<code>*ptr = 5;</code>	Разыменование указателя и присвоение значения <b>5</b>
<code>a = *ptr;</code>	Разыменование указателя и присвоение значения переменной <b>a</b>
<code>printf("%d", *ptr);</code>	Разыменование указателя и вывод его значения на экран.
<code>arr = (int*)malloc(5 * sizeof(int));</code>	Динамическое выделение блока памяти на <b>5</b> элементов типа <b>int</b> и присвоение его адреса указателю <b>arr</b> .
<code>ptr++;</code>	Увеличение значения указателя на размер его типа. Для динамических массивов — переход указателя к следующему элементу.
<code>ptr--;</code>	Уменьшение значения указателя на размер его типа. Для динамических массивов — переход указателя к предыдущему элементу.
<code>ptr2 = ptr1 + 3;</code>	Присваивает указателю <b>ptr2</b> значение указателя <b>ptr1</b> , увеличенное на <b>3</b> размера его типа, т.е. для типа <b>int</b> на 12 байт.
<code>ptr2 = ptr1 - 7;</code>	Присваивает указателю <b>ptr2</b> значение указателя <b>ptr1</b> , уменьшенное на <b>7</b> размеров его типа, т.е. для типа <b>int</b> на 28 байт.
<code>n = ptr1 - ptr2;</code>	Вычисляет разницу между значениями указателей в байтах и присваивает это значение переменной <b>n</b> . Операция сложения для двух указателей смысла не имеет.
<code>free(arr);</code>	Освобождение ранее выделенной памяти.

Рисунок 1 – иллюстрация таблицы основных операций и функций

Ход работы:

Задания для самостоятельной работы:

Задание 1. Указатели и адреса.

Для выполнения задания 1 (приложение А) использовался вариант 12, представленный в лабораторной работе №8 и интегрированная среда разработки Microsoft Visual Studio.

Задание 2. Решить задачи с использованием указателей, динамической памяти и функций. В виде функций оформить ввод, вывод и обработку массива согласно варианту. В main должно быть только объявление массива, выделение динамической памяти и вызов объявленных функций.

Для выполнения задания 2 (приложение Б) использовался вариант 12, представленный в лабораторной работе №8 и интегрированная среда разработки Microsoft Visual Studio.

Задание 3. Решить задачи с использованием указателей, динамической памяти и функций. В виде функций оформить ввод, вывод и обработку массива согласно номеру варианта. В main должно быть только объявление массива, выделение динамической памяти и вызов объявленных функций.

Для выполнения задания 3 (приложение В) использовался вариант 12, представленный в лабораторной работе №8 и интегрированная среда разработки Microsoft Visual Studio.

Вывод: в результате выполнения лабораторной работы получены навыки работы с указателями и динамическими массивами.

## Приложение А

(обязательное)

### Код программы задания 1

```
#include <iostream>
using namespace std;
double a, b;
double* ac = &a;
double* bc = &b;
int main()
{
    /*Ввести значение 2-х вещественных переменных a и b. Направить два указателя на
эти переменные. Затем поменять местами значения
переменных a и b через их указатели*/
    cin >> a >> b;
    cout << a << ' ' << b << endl;
    swap(*ac, *bc);
    cout << a << ' ' << b << endl;
}
```

## Приложение Б

(обязательное)

### Код программы задания 2

```
#include<iostream>
#include<ctime>
using namespace std;
int n = 4;
int pcounter, qcounter;

int* createArray(int n);
void printarr(int* arr);
void compare(int* arr1, int* arr2);
int main()
{
    srand(time(NULL));
    /*Где больше ненулевых элементов*/
    int* q = createArray(n);
    printarr(q);
    int* p = createArray(n);
    printarr(p);
    compare(q, p);
}
int* createArray(int n)
{
    int* arr = new int[n];
    for (int i = 0; i < n; i++)
    {
        arr[i] = -1 + rand() % 5;
    }
    return arr;
}
void printarr(int* arr)
{
    for (int i = 0; i < n; i++)
    {
        cout << arr[i] << " ";
    }
    cout << endl;
}
void compare(int* arr1, int* arr2) {
    for (int i = 0; i < sizeof(arr1); i++) {
        if (arr1[i] != 0) {
            qcounter++;
        }
    }
    for (int i = 0; i < sizeof(arr2); i++) {
        if (arr2[i] != 0) {
            pcounter++;
        }
    }
    if (qcounter > pcounter) {
        cout << "q > p" << endl;
    }
    else if (qcounter < pcounter) {
        cout << "q < p" << endl;
    }
    else {
        cout << "q = p" << endl;
    }
}
```

## Приложение В

(обязательное)

### Код программы задания 3

```
#include <iostream>
#include <ctime>
using namespace std;
int n = 5, m = 5;
int** arr;
int** CreateTwoSArr(int n, int m);
int* CreateOneSArr(int n);
int* FilloSArr(int* fillingArr, int** source);
void PrintTSArr(int** arr);
void PrintOSArr(int* arr);

int main()
{
    srand(time(NULL));
    setlocale(LC_ALL, "Rus");
    int** a = CreateTwoSArr(n, m);
    int* b = CreateOneSArr(m);
    PrintTSArr(a);
    PrintOSArr(b);
    FilloSArr(b, a);
    PrintOSArr(b);
}

int** CreateTwoSArr(int n, int m) {
    int** arr = new int* [n];
    for (int i = 0; i < n; i++) {
        arr[i] = new int[m];
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            arr[i][j] = -10 + rand() % (10 - (-10) + 1);
        }
    }
    return arr;
}

int* CreateOneSArr(int n) {
    int* arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = 0;
    }
    return arr;
}

int* FilloSArr(int* fillingArr, int** source) {
    for (int i = 0; i < m; i++) {
        int sum = 0;
        for (int j = 0; j < n; j++) {
            if(source[j][i] < 0)
                sum += source[j][i];
        }
        fillingArr[i] = sum;
    }
    return fillingArr;
}

void PrintTSArr(int** arr) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cout << arr[i][j] << " ";
        }
        cout << endl;
    }
}
```

```
        cout << endl;
    }
    void PrintOSArr(int* arr) {
        for (int i = 0; i < m; i++) {
            cout << arr[i] << " ";
        }
        cout << endl;
    }
}
```