**Федеральное государственное автономное образовательное учреждение высшего образования**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**(ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАИНДУСТРИИ)**

**(Факультет информационных технологий)**

***(Институт Принтмедиа и информационных технологий)***

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА № 7**

**Дисциплина:** Введение в программирование

**Тема:** Указатели

**Выполнил: студент группы 231-338**

Шаура Илья Максимович

**Дата, подпись** 20.11.2023

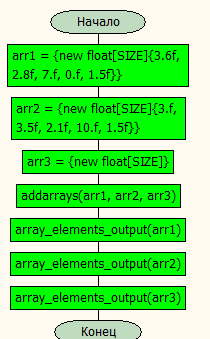
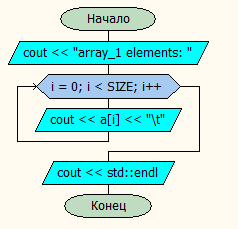


**Москва2023**

Меню выбора лабораторной / пункта



1. Предположим, что в функции main() определены три локальных массива одинакового размера и типа (скажем, float). Первые два уже инициализированы значениями. Напишите функцию addarrays(), которая принимает в качестве аргументов адреса грех массивов, складывает соответствующие элементы двух массивов и помещает результат в третий массив. Четвертым аргументом этой функции может быть размерность массивов. На всем протяжении программы используйте указатели.



void array\_elements\_output(float\* a) {

std::cout << "array\_1 elements: ";

for (size\_t i = 0; i < SIZE; i++)

{

std::cout << a[i] << "\t";

}

std::cout << std::endl;

}

void lab1() {

float\* arr1 = { new float[SIZE] { 3.6f, 2.8f, 7.f, 0.f, 1.5f } };

float\* arr2 = { new float[SIZE] { 3.f, 3.5f, 2.1f, 10.f, 1.5f } };

float\* arr3 = { new float[SIZE] };

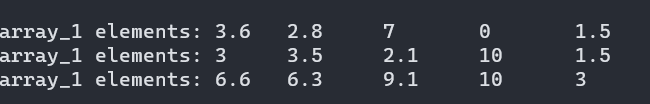
addarrays(arr1, arr2, arr3);

array\_elements\_output(arr1);

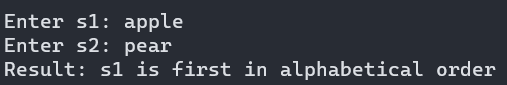
array\_elements\_output(arr2);

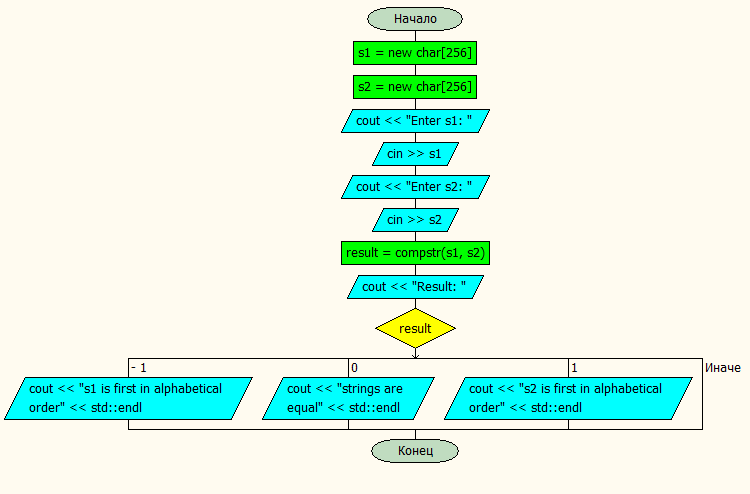
array\_elements\_output(arr3);

}



1. Создайте свою версию библиотечной функции strcmp(s1, s2), которая сравнивает две строки и возвращает -1, если s1 идет первой по алфавиту, 0, если в s1 и s2 одинаковые значения, и 1, если s2 идет первой по алфавиту. Назовите вашу функцию compstr(). Она должна принимать в качестве аргументов два указателя на строки char \*, сравнивать эти строки посимвольно и возвращать число int. Напишите функцию main() для проверки работы вашей функции с разными строками. Используйте указатели во всех возможных ситуациях.





void lab2() {

char\* s1 = new char[256];

char\* s2 = new char[256];

std::cout << "Enter s1: ";

std::cin >> s1;

std::cout << "Enter s2: ";

std::cin >> s2;

int result = compstr(s1, s2);

std::cout << "Result: ";

switch (result)

{

case -1:

std::cout << "s1 is first in alphabetical order" << std::endl;

break;

case 0:

std::cout << "strings are equal" << std::endl;

break;

case 1:

std::cout << "s2 is first in alphabetical order" << std::endl;

break;

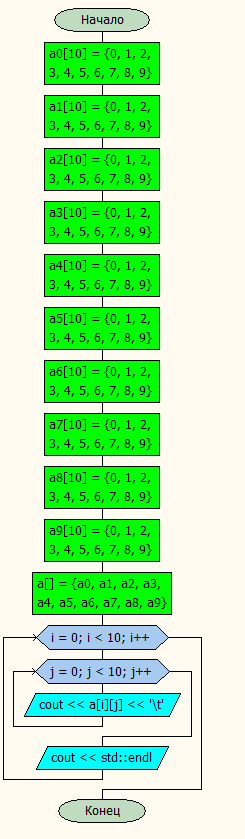
default:

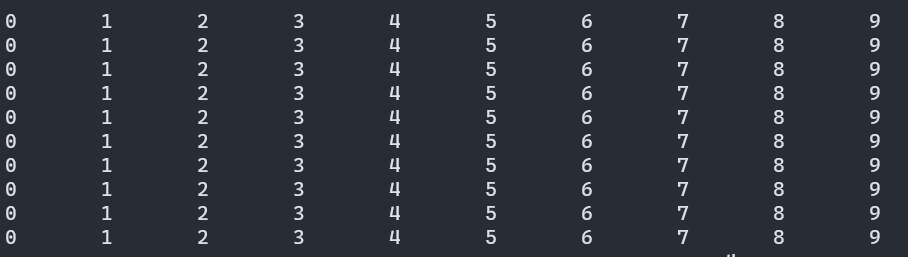
break;

}

}

1. Допустим, что нам нужно сохранить 100 целых чисел так, чтобы иметь к ним легкий доступ. Допустим, что при этом у нас есть проблема: память нашего компьютера так фрагментирована, что может хранить массив, наибольшее количество элементов в котором равно десяти (такие проблемы действительно появляются, хотя обычно это происходит с объектами, занимающими большое количество памяти). Вы можете решить эту проблему, определив 10 разных массивов по 10 элементов в каждом и массив из 10 указателей на эти массивы. Массивы будут иметь имена а0, a1, а2 и т. д. Адрес каждого массива будет сохранен в массиве указателей типа int\*, который называется ар. Вы сможете получить доступ к отдельному целому используя выражение ap[j] [к], где j является номером элемента массива указателей, а к — номером элемента в массиве, на который этот указатель указывает. Это похоже на двумерный массив, но в действительности является группой одномерных массивов.





void lab3() {

int a0[10]{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

int a1[10]{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

int a2[10]{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

int a3[10]{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

int a4[10]{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

int a5[10]{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

int a6[10]{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

int a7[10]{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

int a8[10]{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

int a9[10]{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

int\* a[]{ a0, a1, a2, a3, a4, a5, a6, a7, a8, a9 };

for (size\_t i = 0; i < 10; i++)

{

for (size\_t j = 0; j < 10; j++)

{

std::cout << a[i][j] << '\t';

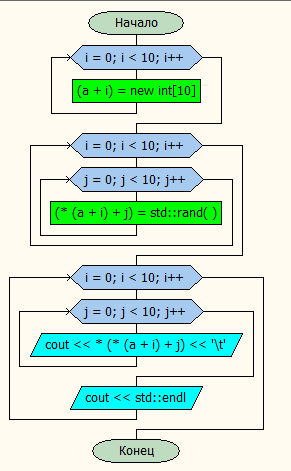
}

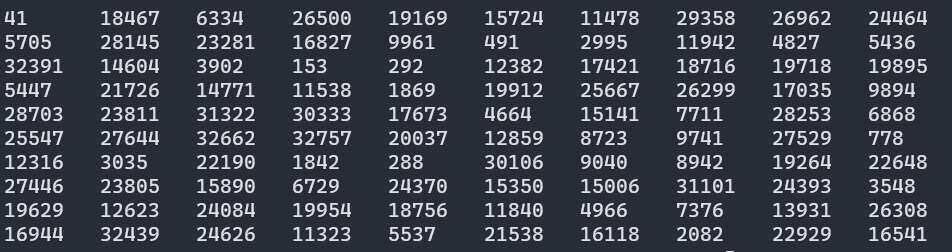
std::cout << std::endl;

}

}

1. Описанный в упражнении 3 подход нерационален, так как каждый из 10 массивов объявляется отдельно, с использованием отдельного имени, и каждый адрес получают отдельно. Вы можете упростить программу, используя операцию new, которая позволит вам выделить память для массивов в цикле и одновременно связать с ними указатели:





void lab4() {

int\* a[10]{};

for (size\_t i = 0; i < 10; i++)

{

\*(a + i) = new int[10];

}

for (size\_t i = 0; i < 10; i++)

{

for (size\_t j = 0; j < 10; j++)

{

\*(\*(a + i) + j) = std::rand();

}

}

for (size\_t i = 0; i < 10; i++)

{

for (size\_t j = 0; j < 10; j++)

{

std::cout << \*(\*(a + i) + j) << '\t';

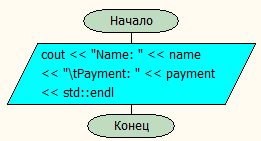
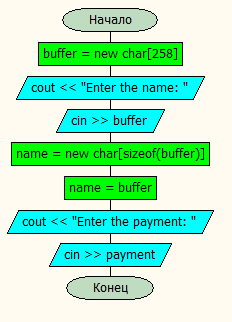
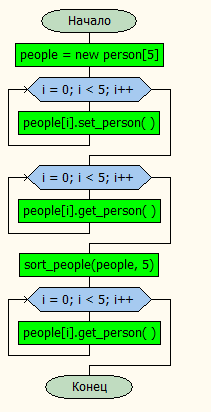
}

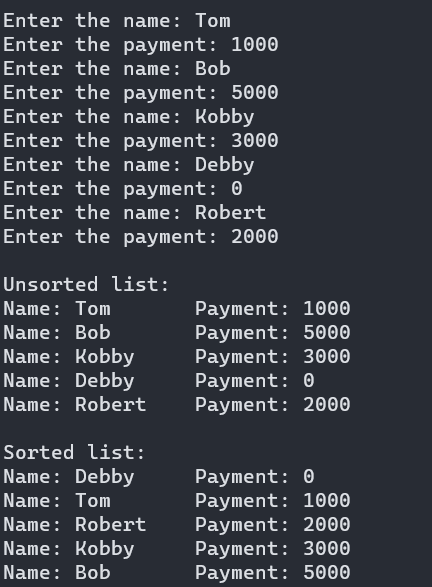
std::cout << std::endl;

}

}

1. Создайте структуру person хранящую информацию об имени и зарплате сотрудников. Для хранения имени использовать указатель на данные типа char. Создайте указатель на данные типа person. Выделите память оператором new для хранения информации о пяти сотрудниках и сохраните адрес ее начала в указателе. Создайте функции для ввода данных о сотрудниках (для хранения имени выделять память динамически и сохранять указатель на нее в соответствующем поле структуры), выводе данных о сотрудниках на экран, сортировки указателей в динамической памяти на сотрудников по величине зарплаты. Проверить работу программы последовательностью операций: Ввод информации о пяти сотрудниках, вывод информации о них на экран, сортировка, вывод информации о сотрудниках на экран.





struct person {

void set\_person() {

char\* buffer = new char[258];

std::cout << "Enter the name: ";

std::cin >> buffer;

name = new char[sizeof(buffer)];

name = buffer;

std::cout << "Enter the payment: ";

std::cin >> payment;

}

void get\_person() {

std::cout << "Name: " << name << "\tPayment: " << payment << std::endl;

}

char\* name = nullptr;

float payment = 0;

};

void lab5() {

person\* people = new person[5];

for (size\_t i = 0; i < 5; i++)

{

people[i].set\_person();

}

std::cout << "\nUnsorted list:" << std::endl;

for (size\_t i = 0; i < 5; i++)

{

people[i].get\_person();

}

sort\_people(people, 5);

std::cout << "\nSorted list:" << std::endl;

for (size\_t i = 0; i < 5; i++)

{

people[i].get\_person();

}

}