**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(а): студент(ка) группы 221-377**

Костоваров А.С.

****** (Фамилия И.О.)

**Дата, подпись** 30.09.2022 ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Проверил: \_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_***

(Фамилия И.О., степень, звание) **(Оценка)**

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва2022**

**Тема**

**Цель:** Получить практические навыки в использовании указателей

.

# Постановка задачи

1. Разработать программы, реализующие задания, приводимые ниже.
2. Оформить отчет, содержащий следующие пункты:
   1. Титульный лист.
   2. Название и цель работы.
   3. Постановка задачи.
   4. **Блок-схемы алгоритмов** и листинги программ с комментариями.

**Код**

#include <iostream>

#include <cstring>

#include <Windows.h>

using namespace std;

struct task1 {

void addarrays(float\* mas1, float\* mas2, float\* mas3, int a) {

for (int i = 0; i < a; i++) {

mas3[i] = mas1[i] + mas2[i];

cout << "Сумма в " << i + 1 << " ячейке: " << mas3[i] << endl;

}

}

void task\_main() {

float mas1[2] = { 2, 3 };

float mas2[2] = { 6, 7 };

float mas3[2];

addarrays(mas1, mas2, mas3, 2);

}

};

struct task2 {

int CompStr(char\* s1, char\* s2)

{

for (int i = 0; (s1[i] != '\0') || (s2[i] != '\0'); i++)

{

if (s1[i] > s2[i])

return 1;

else if (s2[i] > s1[i])

return -1;

}

return 0;

}

void task\_main()

{

char\* str1 = new char[80], \* str2 = new char[80];

char prob = '/';

cout << "Введите две строки(через enter,eng), которые хотите сравнить" << endl;

cout << "Если вторая строка идет первее по алфавиту, программа вернет 1" << endl;

cout << "Если первая строка идет первее по алфавиту, программа вернет -1" << endl;

cout << "Если равны, программа вернет 0" << endl;

cin >> str1;

cin >> str2;

cout << CompStr(str1, str2);

}

};

struct task3 {

void task\_main() {

const int s = 10;

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

int a1[] = { 10,11,12,13,14,15,16,17,18,19 };

int a2[] = { 20,21,22,23,24,25,26,27,28,29 };

int a3[] = { 30,31,32,33,34,35,36,37,38,39 };

int a4[] = { 40,41,42,43,44,45,46,47,48,49 };

int a5[] = { 50,51,52,53,54,55,56,57,58,59 };

int a6[] = { 60,61,62,63,64,65,66,67,68,69 };

int a7[] = { 70,71,72,73,74,75,76,77,78,79 };

int a8[] = { 80,81,82,83,84,85,86,87,88,89 };

int a9[] = { 90,91,92,93,94,95,96,97,98,99 };

int\*\* ap = new int\* [s];

ap[0] = a0;

ap[1] = a1;

ap[2] = a2;

ap[3] = a3;

ap[4] = a4;

ap[5] = a5;

ap[6] = a6;

ap[7] = a7;

ap[8] = a8;

ap[9] = a9;

cout << "массив: \n";

for (int i = 0; i < s; i++) {

for (int k = 0; k < s; k++) {

cout << " " << ap[i][k];

}

cout << endl;

}

}

};

struct task4 {

void task\_main() {

setlocale(LC\_ALL, "Russian");

const int MAXSIZE = 10;

int\* ap[MAXSIZE];

int NUMARRAYS = 10, m = 0;

for (int j = 0; j < MAXSIZE; j++) {

\*(ap + j) = new int[MAXSIZE];

}

cout << "массив: \n";

for (int j = 0; j < 10; j++) {

for (int k = 0; k < 10; k++) {

\*(\*(ap + j) + k) = m++;

cout << " " << \*(\*(ap + j) + k);

}

cout << endl;

}

}

};

struct task5 {

struct Person

{

int salary = 0;

char\* pName = new char[20];

Person()

{

}

Person(char\* h, int sala)

{

SetData(h, sala);

}

void GetData()

{

cout << "Salary: " << salary << endl;

cout << "Name: ";

for (int i = 0; \*(pName + i) != '\0'; i++)

{

cout << \*(pName + i);

}

cout << endl;

}

void SetData(char\* h, int sal)

{

pName = h;

salary = sal;

}

};

void Sort(Person\* pArr, int size)

{

int i = 0, j = 0;

for (i = 1; i < size; i++)

{

int indBuf = i;

Person buf = \*(pArr + i);

for (j = i; (j > 0) && (pArr[j - 1].salary < buf.salary); j--)

{

pArr[j] = pArr[j - 1];

}

pArr[j] = buf;

}

}

void ArrInfo(Person\* pArr, int size)

{

for (int i = 0; i < size; i++)

{

(pArr + i)->GetData();

}

}

Person InitPerson()

{

cin.ignore();

int s;

char\* h = new char[30];

cout << "Введите имя сотрудника: ";

cin.getline(h, 30);

cout << "Введите зарплату сотрудника: ";

cin >> s;

cin.ignore();

return Person(h, s);

}

void task\_main()

{

setlocale(LC\_ALL, "Russian");

Person\* Pperson = new Person[5];

for (int i = 0; i < 5; i++)

{

\*(Pperson + i) = InitPerson();

}

ArrInfo(Pperson, 5);

Sort(Pperson, 5);

cout << "=================================================" << endl;

ArrInfo(Pperson, 5);

}

};

int main() {

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int checknomber;

for (int checknomber = -1; checknomber != 0;) {

switch (checknomber)

{

case 1:

task1 t1;

t1.task\_main();

cout << "" << endl;

break;

case 2:

task2 t2;

t2.task\_main();

cout << "" << endl;

break;

case 3:

task3 t3;

t3.task\_main();

cout << "" << endl;

break;

case 4:

task4 t4;

t4.task\_main();

cout << "" << endl;

break;

case 5:

task5 t5;

t5.task\_main();

cout << "" << endl;

break;

}

cout << "Для окончания работы введите 0\n";

cout << "Введите номер задания: ";

cin >> checknomber;;

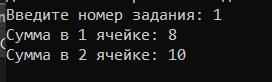
}

}

**Постановка задачи 1**

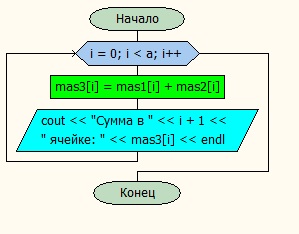
Предположим, что в функции main() определены три локальных массива одинакового размера и типа (скажем, float). Первые два уже инициализированы значениями. Напишите функцию addarrays(), которая принимает в качестве аргументов адреса грех массивов, складывает соответствующие элементы двух массивов и помещает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

**Вывод:**

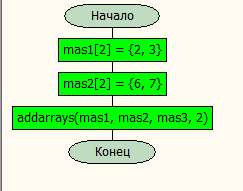
****

**Блок-схема**

**addarrays**

****

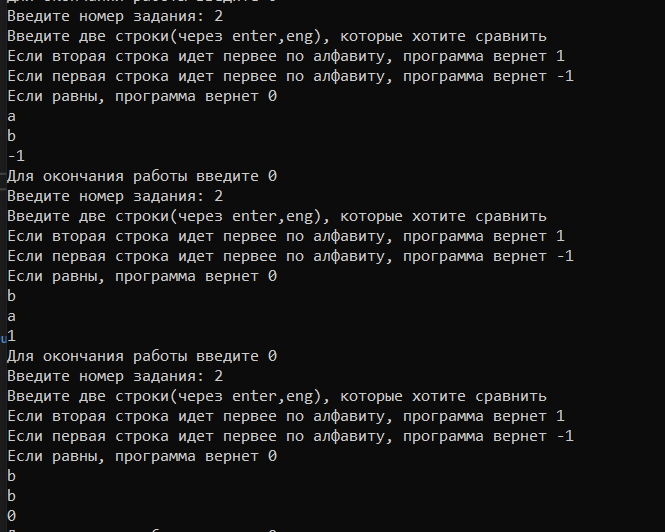
**Task\_main**

****

**Постановка задачи 2**

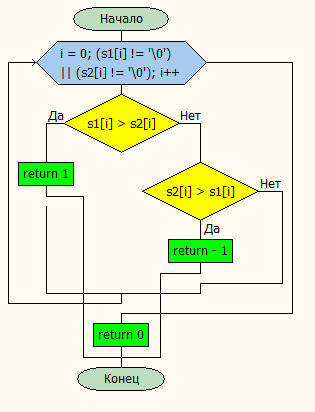
Создайте свою версию библиотечной функции strcmp(s1, s2), которая сравнивает две строки и возвращает -1, если s1 идет первой по алфавиту, 0, если в s1 и s2 одинаковые значения, и 1, если s2 идет первой по алфавиту. Назовите вашу функцию compstr(). Она должна принимать в качестве аргументов два указателя 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.

**Вывод:**

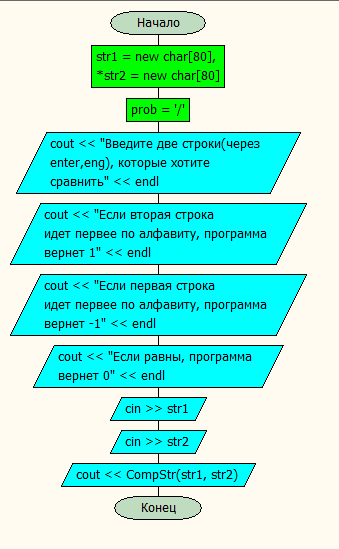
****

**Блок-схема**

**CompStr**

****

**Task\_main**

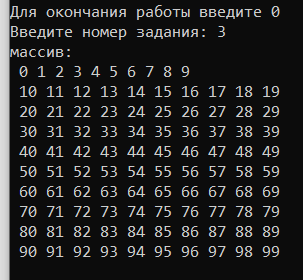
****

**Постановка задачи 3**

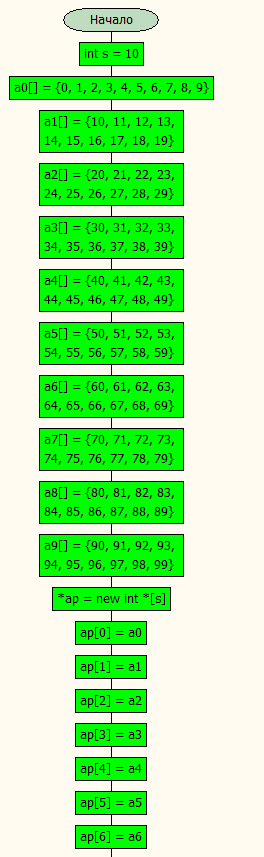
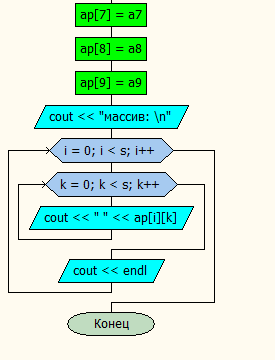
Допустим, что нам нужно сохранить 100 целых чисел так, чтобы иметь к ним легкий доступ. Допустим, что при этом у нас есть проблема: память нашего компьютера так фрагментирована, что может хранить массив, наибольшее количество элементов в котором равно десяти (такие проблемы действительно появляются, 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.

Заполните группу массивов тестовыми данными (скажем, номерами 0, 10, 20 и т. д.), а затем выведите их, чтобы убедиться, что все работает правильно.

**Вывод:**

****

**Блок-схема**

**Постановка задачи 4**

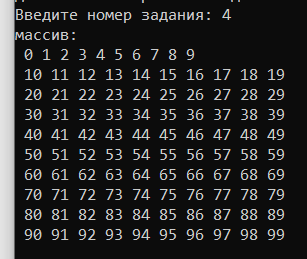
Описанный в упражнении 3 подход нерационален, так как каждый из 10 массивов объявляется отдельно, с использованием отдельного имени, и каждый адрес получают отдельно. Вы можете упростить программу, используя операцию new, которая позволит вам выделить память для массивов в цикле и одновременно связать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

for ( j = 0: j < NUMARRAYS; j++ ) // создаем NUMARRAYS массивов

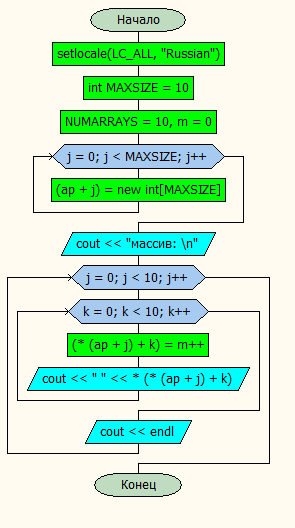
\*( ар + j ) = new int [ MAXSIZE ]: //no MAXSIZE целых чисел в каждом

Перепишите программу упражнения 3, используя этот подход. Доступ к отдельному элементу массивов вы сможете получить, используя то же выражение, что и в упражнении 3, или аналогичное выражение с указателями: \*(\*(ap+j)+k).

**Вывод:**

****

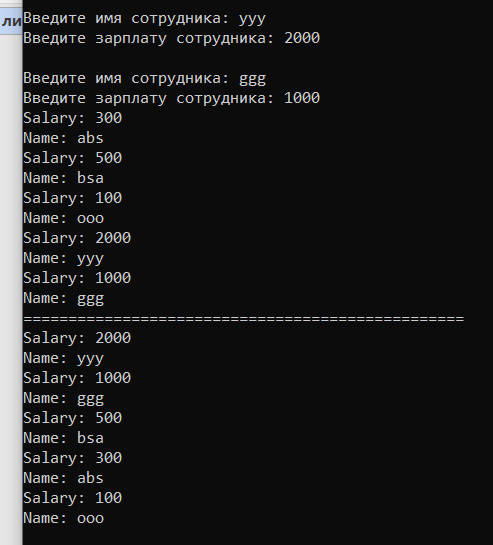
**Блок-схема**

****

**Постановка задачи 5**

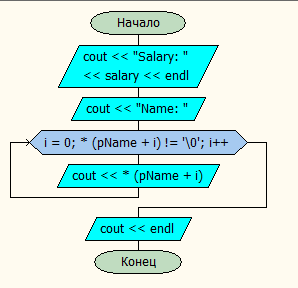
Создайте структуру person хранящую информацию об имени и зарплате сотрудников. Для хранения имени использовать указатель на данные типа char. Создайте указатель на данные типа person. Выделите память оператором new для хранения информации о пяти сотрудниках и сохраните адрес ее начала в указателе. Создайте 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.

**Вывод:**

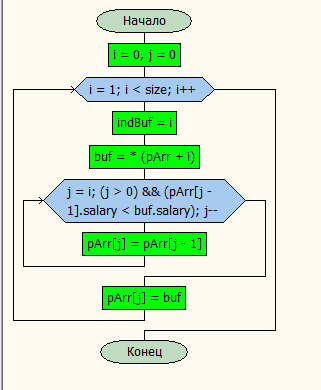
****

**Блок-схема**

**Getdata**

****

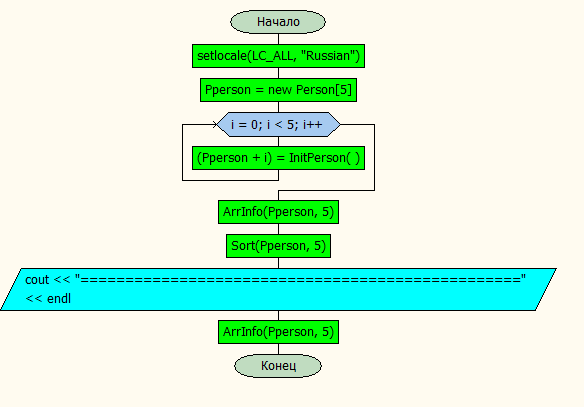
**Sotr**

****

**Intperson**



Task\_main



Main

