Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Восточно-Сибирский государственный университет технологий и управления

(ФГБОУ ВСГУТУ)

Электротехнический факультет

Кафедра «Системы Информатики»

Дисциплина

«Программирование»

Отчет по лабораторной работе №1

На тему:

Пользовательские типы данных

Вариант7

|  |  |
| --- | --- |
| Выполнил: | студент гр. Б661-2б |
|  | Немеров А.П. |
| Проверил: | преподаватель  Жамбалов Э.Б. |

Улан-Удэ

2021

**Общая задача №1**

**1 Словесная постановка задачи**

Разработать программу, создать в программе структуру, в которой определены координаты точки в трехмерном пространстве (x, y, z). Обеспечить ввод массива таких точек. Вывести список точек и их координат на экран, а также определить точку, которая ближе всех к началу координат.

**1.1 Решение задачи**

**1.2 Формальная постановка задачи**

Входные данные:

coordinates – структура содержащая переменные с координатами;

x, y, z - целые числа, представляющие координаты точек;

input – целое число, представляющее кол-во точек;

Выходные данные:

**Структура координат:** Эти данные объявляются так:

coordinates \*xyz = new coordinates[input];

**Расстояние от начала координат до точки:**

Эти данные объявляются так:

Double Pif;

**Искомая точка:** Эти данные объявляются так:

Int maxI= 0;

**Минимальное расстояние от начала координат до точки:**

Эти данные объявляются так:

double minPif = 10000000;

Метод получения значений вычисляемых полей:

double Pif = sqrt(xyz[i].x \* xyz[i].x + xyz[i].y \* xyz[i].y + xyz[i].z \* xyz[i].z);

**1.3 Реализация**

В функции main выводится меню с выбором задания, общее 1-2, индивидуальное и дополнительное, соответственно функции firstGeneralTask, secondGeneralTask, individualTask и additionalTask.

Перед реализацией firstGeneralTask объявлена структура координат, содержащая целочисленные переменные x, y, z. Функция firstGeneralTask начинается с ввода количества точек и инициализации массива структур, целочисленных переменных минимального расстояния от начала координат до точки и переменная для хранения номера точки. Затем цикл записи координат точек и цикл вычислений всех расстояний всех точек и поиск минимального расстояния, и вывод всех точек и координат. После цикла вывод полученных данных.

#pragma region firstGeneralTask

Структура координат

struct coordinates {

int x;

int y;

int z;

};

Основная функция

void firstGeneralTask() {

Ввод количества точек

cout << "\n\n\tВведите количество точек: ";

int input;

cin >> input;

if (input == 0) { return; }

double minPif = 10000000000;

Инициализация массива структур

coordinates\* xyz = new coordinates[input];

int maxI = 0;

system("cls");

Цикл заполнения массива структур

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

cout << "\n\n\tВведите координаты точки " << i + 1;

cout << "\n\n\tВведите x: "; cin>>xyz[i].x;

cout << "\n\tВведите у: "; cin>>xyz[i].y;

cout << "\n\tВведите z: "; cin>>xyz[i].z;

system("cls");

}

Цикл вычисления расстояния от начала координат до точки

И поиска минимально отдаленного от начала координат

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

double Pif = sqrt(xyz[i].x \* xyz[i].x + xyz[i].y \* xyz[i].y + xyz[i].z \* xyz[i].z);

if (minPif > Pif) {

minPif = Pif;

maxI = i;

}

Вывод полученных значений

cout << "\n\t" << i+1 << ".\tточка с координатами {" << xyz[i].x << " , " << xyz[i].y << " , " << xyz[i].z << "}";

}

cout << "\n\n\tНаиближайщая точка к началам координат " << maxI+1 << " с координатами {" << xyz[maxI].x << " , " << xyz[maxI].y << " , " << xyz[maxI].z << "}";

cout << "\n\n\tРастояние от начала координат = " << minPif;

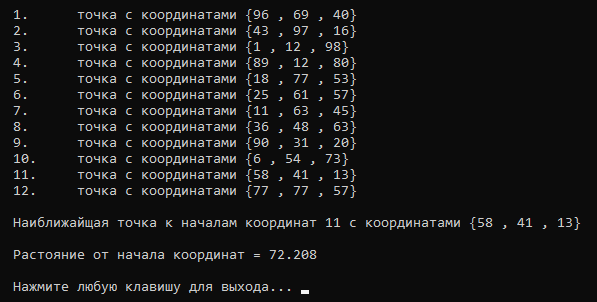
cout << "\n\n\tНажмите любую клавишу для выхода... "; char p = \_getch(); system("cls"); return;

}

#pragma endregion

**1.4 Тестирование**

Работа в штатном режиме, введём валидные данные:



**Общая задача №2**

**2 Словесная постановка задачи**

Разработать программу, с помощью структуры, битового поля и объединения, создайте тип данных, позволяющий переводить целое число из десятичной в двоичную форму без каких-либо операций, кроме присваивания и вывода на экран.

**2.1 Решение задачи**

**2.2 Формальная постановка задачи**

Входные данные:

BITSUNION – объединение, содержащее целочисленную переменную и битовое поле

x - целое число;

BitsStruct – структура состоящая из битовых полей;

Выходные данные:

**Объединение:** Эти данные объявляются так:

BITSUNION Buni;

**Структура(Битовые поля):** Эти данные объявляются так:

Struct BitsStruct{

b1:1;

b2:1;

…….

b16:1;

}res;

**2.3 Реализация**

Перед реализацией secondGeneralTask объявлено объединение состоящее из целочисленной переменной и битового поля. В самой функции secondGeneralTask вводится целочисленная переменная в объединение, а затем выводится из объединения через битовое поле.

#pragma region secondGeneralTask

Объявляем объединение

union BITSUNION{

int x;

Объявляем структуру с битовыми полями размеров в 16 бит

struct BitsStruct{

unsigned short int b1 : 1;

unsigned short int b2 : 1;

unsigned short int b3 : 1;

unsigned short int b4 : 1;

unsigned short int b5 : 1;

unsigned short int b6 : 1;

unsigned short int b7 : 1;

unsigned short int b8 : 1;

unsigned short int b9 : 1;

unsigned short int b10 : 1;

unsigned short int b11 : 1;

unsigned short int b12 : 1;

unsigned short int b13 : 1;

unsigned short int b14 : 1;

unsigned short int b15 : 1;

unsigned short int b16 : 1;

}res; Инициализируем структуру с битовыми полями

};

void secondGeneralTask() {

BITSUNION Buni; Инициализируем объединение

cout << "\n\n\tВведите число: ";

cin >> Buni.x; Ввод числа

system("cls");

В ывод числа в двоичной системе счисления

cout << "\n\n\tВведенное число в двоичной системе счисления: "

<< Buni.res.b16 << Buni.res.b15

<< Buni.res.b14 << Buni.res.b13

<< Buni.res.b12 << Buni.res.b11

<< Buni.res.b10 << Buni.res.b9

<< Buni.res.b8 << Buni.res.b7

<< Buni.res.b6 << Buni.res.b5

<< Buni.res.b4 << Buni.res.b3

<< Buni.res.b2 << Buni.res.b1 << endl;

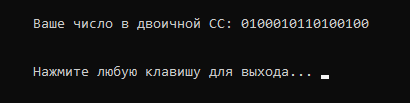
cout << "\n\n\tНажмите любую клавишу для выхода... "; char p = \_getch(); system("cls"); return;

}

#pragma endregion

**2.4 Тестирование**

Работа в штатном режиме, введём валидные данные: 542116



**Индивидуальная задача**

**3 Словесная постановка задачи**

Разработать программу, с помощью структуры «Покупатель» ( Состоящая из: Даты покупки; ФИО; домашнего адреса; номера телефона; суммы покупки). Вычисляемое значение – общая сумма покупки.

В программе реализовать меню и соответствующие функции:

1) Создание нового элемента массива структур (ввод его данных);

2) Сортировка массива структур;

3) Изменение заданной структуры (вводится номер элемента в массиве структур, и его данные изменяются на новые введенные пользователем данные);

4) Удаление структуры из массива (можно удаляемому элементу массива присвоить значение последнего элемента и уменьшить количество элементов в массиве структур);

5) Вывод на экран массива структур в виде таблицы. Также выводите вычисляемое значение по всему массиву, указанное в вашем варианте.

6) Поиск в массиве структур по заданному параметру (самостоятельно выберите поле структуры и реализуйте поиск по нему).

**3.1 Решение задачи**

**3.2 Формальная постановка задачи**

Входные данные:

person – структура, содержащее данные клиента (

string Surname Фамилию

string Name Имя

string Patronymic Отчество

string Address Адрес

string PhoneNum Телефонный номер

string PayDay день оплаты

int Summ сумму оплаты )

maxSize - целое число, содержащее максимальное количество клиентов;

counter – целое число, содержащее количество клиентов на данный момент;

inputNum, InInt, inputH, inP - целое число, содержащее номер пункта;

surnames, names, Patronymics, Address, PhoneNums, PayDays – строковые массивы, содержащие данные первых 10 клиентов;

newSurname, newName, newPatronymic, newAddress, newPhoneNum, newPayDay, newSumm – переменные для записи новых клиентов;

temp – временная строковая переменная;

tempInt – временная целочисленная переменная;

inputStr – строковая переменная для поиска по ней;

maxLenSurname, maxLenName, maxLenPatronymic, maxLenAddress, maxLenPhoneNum, maxLenPayDay, maxLenSumm - целые числа, представляющие собой максимальную длину соответственных данных структуры;

LenSurname, LenName, LenPatronymic, LenAddress, LenPhoneNum, LenPayDay, LenSumm – целые числа, представляющие собой длину соответственных данных структуры;

Выходные данные:

**Переменная найденного клиента:** Эти данные объявляются так:

searchID = -1;

**Структура:** Эти данные объявляются так:

struct person {

string Surname = { "None" };

string Name = { "None" };

string Patronymic = { "None" };

string Address = { "None" };

string PhoneNum = { "None" };

string PayDay = { "None" };

int Summ = 0;

};

**3.3 Реализация**

Перед основной функцией (individualTask) объявляется структура с персональными данными клиента (Фамилия string Surname, Имя string Name, Отчество(при наличии, иначе None) string Patronymic, Домашний адрес string Address, Телефонный номер string PhoneNum, День оплаты string PayDay, Сумма оплата int Summ). После экземпляры основных функций. Затем идёт главная функция individualTask, сначала ввод максимального количества клиентов и инициализация массива структур, после вывод меню с выбором пунктов-функций

fillClients просто функция для заполнения первых 10 клиентов

addClients объявление переменных и ввод, и присвоение в конец массива структур

swapClients функция для того чтобы менять местами данные клиента

sortClients функция сортировки, сначала спрашивается по какому полю будет производится поиск (по Фамилии или Имени), а после обычная сортировка «пузырьком», сравнение двух слов. При сортировке используется функция swapClients. Таким образом сортируется по алфавиту.

transform в функции сначала выводится меню с выбором, какие именно данные клиента нужно изменить, потом ввод пункта меню и ввод изменения и перезапись

transformClients функция сначала выводит по какому полям искать клиента для изменения, поиск по ФИО это сравнение введенной строки с полями Фамилия, Имя и Отчество, найденный номер в таблице отправляется в функцию transform. А изменение по номеру в таблице просто уточняется точно его изменить или отмена.

swapAndDelClients функция по удалению данных клиента, на его место записывается следующий по номеру клиент, идущие после удаленного клиента смещаются «выше» на один номер. Размер клиентов уменьшается на одну единицу.

delClients функция удаления данных клиента, сначала аналогичный поиск как и в функции transformClients, только найденный номер клиента передаётся в функцию swapAndDelClients

showClients функция вывода всех данных клиентов в виде таблицы, сначала объявление переменных, которые содержат в себе размер каждого столбца, изначально в этих переменных указаны минимальные размеры, в качестве минимальных значений указаны длины заголовков столбцов. Дальше идет поиск максимально длинного в столбце и запись его длины в переменную содержащую максимальную длину столбца, если длина больше изначальной. Потом вывод данных в виде таблице с получившимися измерениями

show функция которая выводит данные найденного клиента.

searchClients функция поиска по ФИО, поиск осуществляется по всем трём столбцам и сразу же найденный клиент, его найденный номер отправляется в функцию show.

**#pragma region individualTask**

**Объявление структуры**

**struct person {**

**string Surname = { "None" };**

**string Name = { "None" };**

**string Patronymic = { "None" };**

**string Address = { "None" };**

**string PhoneNum = { "None" };**

**string PayDay = { "None" };**

**int Summ = 0;**

**};**

**Прототипы функций по пунктам меню**

**void fillClients (person\* clients, int\* counter);**

**void addClients (person\* clients, int\* counter);**

**void sortClients (person\* clients, int\* counter);**

**void transformClients (person\* clients, int\* counter);**

**void delClients (person\* clients, int\* counter);**

**void showClients (person\* clients, int\* counter);**

**void searchClients (person\* clients, int\* counter);**

**Основная функция**

**void individualTask() { //<<<<--------main индивидуального задания**

**int maxSize;**

**system("cls");**

**cout << "\n\tВведите максимальное количество клиентов: ";**

**cin >> maxSize;**

**Счётчик кол-ва клиентов на данный момент**

**int \*counter = new int;**

**\*counter = 0;**

**Инициализация структуры**

**person \*clients = new person[maxSize];**

**system("cls");**

**функция для заполнения первых 10 клиентов**

**fillClients(clients, counter);**

**/////// МЕНЮ /////////**

**while (true) {**

**std::cout << "\n\t \_\_\_ \_\_\_ \_\_\_\_\_\_ \_\_ \_\_\_ \_\_ \_\_";**

**std::cout << "\n\t /' \_\_` \_\_`\\ /\\ \_\_\_\\ /\\ \\/ \_\_`\\ /\\ \\ / \\";**

**std::cout << "\n\t /\\ \\/\\ \\/\\ \\ \\ \\ \_\_\_\\ \\ \\ /\_ \\ \\ \\ \\ \\/ /\\ \\";**

**std::cout << "\n\t \\ \\\_\\ \\\_\\ \\ \\ \\ \\\_\_\_\_\_\\ \\ \\\_\\ \\ \\\_\\ \\ \\\_\_/\\ \\\_\\";**

**std::cout << "\n\t \\/\_/\\/\_/\\/\_/ \\/\_\_\_\_\_/ \\/\_/ \\/\_/ \\/\_\_/ \\/\_/\n";**

**std::cout << "\n\t ----== Введите номер нужного действия ==---- ";**

**std::cout << "\n\t ----== 1. Добавить клиента ==---- ";**

**std::cout << "\n\t ----== 2. Сортировать всех клиентов ==---- ";**

**std::cout << "\n\t ----== 3. Изменить данные клиента ==---- ";**

**std::cout << "\n\t ----== 4. Удалить данные клиента ==---- ";**

**std::cout << "\n\t ----== 5. Вывести список клиентов ==---- ";**

**std::cout << "\n\t ----== 6. Поиск клиента по базе ==---- ";**

**//std::cout << "\n\t ----== 7. Выборка клиентов по сумме ==---- ";**

**std::cout << "\n\t ----== 0. Выход ==---- \n\t";**

**int inputNum;**

**std::cin >> inputNum;**

**std::system("cls");**

**switch (inputNum) {**

**case(0): {std::cout << "\n\tВыход\n"; return; }**

**case(1): {addClients (clients, counter); break; }**

**case(2): {sortClients (clients, counter); break; }**

**case(3): {transformClients (clients, counter); break; }**

**case(4): {delClients (clients, counter); break; }**

**case(5): {showClients (clients, counter); break; }**

**case(6): {searchClients (clients, counter); break; }**

**//case(7): {selectionClients (clients, counter); break; }**

**default: {std::cout << "\n\tВведено неверное значение, нажмите любую клавишу для продолжения... "; char p = \_getch(); std::system("cls"); break; }**

**}**

**}**

**}**

**Функция заполнения 10 первых клиентов**

**void fillClients(person\* clients, int\* counter) {**

**string surnames[10] = { "Wallace", "Сафонова", "Виноградов", "Зотова", "Шапошников", "Brandon", "Дементьев", "Корнев", "Пономарев", "Ефремова" };**

**string names[10] = { "Carter", "Мария", "Олег", "Анастасия", "Макар", "Rose", "Арсен", "Лука", "Кирилл", "Анна" };**

**string Patronymics[10] = { "None", "Егоровна", "Андреевич", "Степановна", "Михайлович", "None", "Миронович", "Сергеевич", "Константинович", "Данииловна", };**

**string Address[10] = {**

**"3959 Wyatt Street, West Palm Beach, Florida",**

**"447191, Саратовская область, город Павловский Посад, въезд Гоголя, 92",**

**"737989, Тульская область, город Раменское, бульвар 1905 года, 22",**

**"434586, Ульяновская область, город Пушкино, проезд Ладыгина, 59",**

**"337525, Читинская область, город Чехов, наб. Домодедовская, 97",**

**"3850 Cedarstone Drive, Toledo, Ohio",**

**"107943, Липецкая область, город Зарайск, бульвар Славы, 64",**

**"314032, Калининградская область, город Ногинск, шоссе Космонавтов, 85",**

**"038819, Томская область, город Клин, бульвар Косиора, 27",**

**"139377, Орловская область, город Наро-Фоминск, шоссе Славы, 43", };**

**string PhoneNums[10] = { "+18005553535", "+78652158365", "+78664788078", "+78027969659", "+78895542709", "+18655790207", "+78032538250", "+78253336952", "+78412045196", "+78003477306" };**

**string PayDays[10] = { "29.03.2021", "27.02.2021", "03.04.2021", "01.02.2022", "14.12.2021", "23.11.2021", "09.06.2021", "22.05.2021", "01.02.2021", "05.12.2021", };**

**int Summs[10] = { 10333, 20419, 9103, 23374, 1826, 2873, 7059, 5318, 20832, 7256, };**

**for (int i = 0; i < 10; i++) {**

**clients[\*counter].Surname = surnames[i];**

**clients[\*counter].Name = names[i];**

**clients[\*counter].Patronymic = Patronymics[i];**

**clients[\*counter].Address = Address[i];**

**clients[\*counter].PhoneNum = PhoneNums[i];**

**clients[\*counter].PayDay = PayDays[i];**

**clients[\*counter].Summ = Summs[i];**

**(\*counter)++;**

**}**

**}**

**//finished**

**Функция для добавления клиента**

**void addClients(person\* clients, int \*counter) {**

**std::cout << "\n\n\tДобавление клиента\n";**

**string newSurname = {"None"};**

**string newName = { "None" };**

**string newPatronymic = { "None" };**

**string newAddress = { "None" };**

**string newPhoneNum = { "None" };**

**string newPayDay = { "None" };**

**int newSumm = 0;**

**std использовались потому что Visual Studio не хотел работать без std, но потом это само починилось**

**std::cout << "\n\tВведите данные клиента\n";**

**std::cout << "\n\tФамилия: "; cin.get(); std::getline(cin, newSurname);**

**std::cout << "\n\tИмя: "; std::getline(cin, newName);**

**std::cout << "\n\tОтчество: "; std::getline(cin, newPatronymic);**

**std::cout << "\n\tАдресс: "; std::getline(cin, newAddress);**

**std::cout << "\n\tНомер телефона (без +7): "; std::getline(cin, newPhoneNum);**

**std::cout << "\n\tДень оплаты (ДД.ММ.ГГГГ): "; std::getline(cin, newPayDay);**

**std::cout << "\n\tСумму покупки: "; cin>> newSumm;**

**std::system("cls");**

**clients[\*counter].Surname = newSurname;**

**clients[\*counter].Name = newName;**

**clients[\*counter].Patronymic = newPatronymic;**

**clients[\*counter].Address = newAddress;**

**clients[\*counter].PhoneNum = "+7" + newPhoneNum;**

**clients[\*counter].PayDay = newPayDay;**

**clients[\*counter].Summ = newSumm;**

**(\*counter)++;**

**std::cout << "\n\tДобавлен \n" << "\n\tфамилия: " << newSurname**

**<< "\n\tИмя: " << newName << "\n\tОтчество: " << newPatronymic**

**<< "\n\tАдресс: " << newAddress << "\n\tНомер телефона: +7" << newPhoneNum**

**<< "\n\tДень оплаты: " << newPayDay << "\n\tСумму покупки: " << newSumm;**

**std::cout << "\n\n\tНажмите любую клавишу для возврата в меню... "; char p = \_getch(); std::system("cls");**

**std::system("cls");**

**};**

**//finished**

**Подфункция сортировки, по перемещению клиентов**

**void swapClients(person\* clients, int\* counter, int j) {**

**string temp = clients[j].Patronymic;**

**clients[j].Patronymic = clients[j + 1].Patronymic;**

**clients[j + 1].Patronymic = temp;**

**temp = clients[j].Name;**

**clients[j].Name = clients[j + 1].Name;**

**clients[j + 1].Name = temp;**

**temp = clients[j].Surname;**

**clients[j].Surname = clients[j + 1].Surname;**

**clients[j + 1].Surname = temp;**

**temp = clients[j].Address;**

**clients[j].Address = clients[j + 1].Address;**

**clients[j + 1].Address = temp;**

**temp = clients[j].PhoneNum;**

**clients[j].PhoneNum = clients[j + 1].PhoneNum;**

**clients[j + 1].PhoneNum = temp;**

**temp = clients[j].PayDay;**

**clients[j].PayDay = clients[j + 1].PayDay;**

**clients[j + 1].PayDay = temp;**

**int tempInt = clients[j].Summ;**

**clients[j].Summ = clients[j + 1].Summ;**

**clients[j + 1].Summ = tempInt;**

**}**

**Функция сортировки**

**void sortClients(person\* clients, int\* counter) {**

**Обработка исключения**

**if (\*counter == 1) { cout << "\n\n\tВ базу всего лишь один клиент, сортировать нечего, нажмите любую кдавишу для возврата в меню... "; char p = \_getch(); system("cls"); return;**

**}**

**Сортировка просто сравнение двух слов**

**std::cout << "\n\tСортировать всех клиентов по\n";**

**std::cout << "\n\t1. Фамилии\t2. Имя\t0.Выход\n";**

**int inputNum;**

**cin >> inputNum;**

**system("cls");**

**switch (inputNum) {**

**case(0): {cout << "\n\n\tВыход"; system("cls"); return; }**

**case(1): {**

**for (int i = 1; i < \*counter; i++) {**

**for (int j = 0; j < (\*counter) - i; j++) {**

**if (clients[j].Surname > clients[j + 1].Surname) {**

**swapClients(clients, counter, j);**

**}**

**}**

**}**

**cout << "\n\n\tСортировка по фамилии завершена, нажмите любую кдавишу для возврата в меню... "; char p = \_getch(); system("cls"); return;**

**}**

**case(2): {**

**for (int i = 1; i < \*counter; i++) {**

**for (int j = 0; j < (\*counter) - i; j++) {**

**if (clients[j].Name > clients[j + 1].Name) {**

**swapClients(clients, counter, j);**

**}**

**}**

**}**

**cout << "\n\n\tСортировка по имени завершена, нажмите любую кдавишу для возврата в меню... "; char p = \_getch(); system("cls"); return;**

**}**

**default: {cout << "\n\n\tВведено неверное значение, нажмите любую клавишу для возврата в меню... "; char p = \_getch(); system("cls"); break; }**

**}**

**std::system("cls");**

**}**

**//finished**

**Подфункция изменения данных клиента**

**void transform(person\* clients, int\* counter, int searchID) {**

**cout << "\n\n\tКакие данные изменить\n"**

**<< "\n\t1. Фамилию - " << clients[searchID].Surname**

**<< "\n\t2. Имя - " << clients[searchID].Name**

**<< "\n\t3. Отчество - " << clients[searchID].Patronymic**

**<< "\n\t4. Адресс - " << clients[searchID].Address**

**<< "\n\t5. Телефонный номер - " << clients[searchID].PhoneNum**

**<< "\n\t6. День оплаты - " << clients[searchID].PayDay**

**<< "\n\t7. Сумму - " << clients[searchID].Summ**

**<< "\n\t0. Выход\n\n\t";**

**int InInt;**

**cin >> InInt;**

**switch (InInt) {**

**case(0): {break; }**

**case(1): {cout << "\n\n\tВведите Фамилию: "; string temp; cin.get(); getline(cin, temp); clients[searchID].Surname = temp; break; }**

**case(2): {cout << "\n\n\tВведите Имя: "; string temp; cin.get(); getline(cin, temp); clients[searchID].Name = temp; break; }**

**case(3): {cout << "\n\n\tВведите Отчество: "; string temp; cin.get(); getline(cin, temp); clients[searchID].Patronymic = temp; break; }**

**case(4): {cout << "\n\n\tВведите Адресс: "; string temp; cin.get(); getline(cin, temp); clients[searchID].Address = temp; break; }**

**case(5): {cout << "\n\n\tВведите Телефонный: "; string temp; cin.get(); getline(cin, temp); clients[searchID].PhoneNum = temp; break; }**

**case(6): {cout << "\n\n\tВведите День: "; string temp; cin.get(); getline(cin, temp); clients[searchID].PayDay = temp; break; }**

**case(7): {cout << "\n\n\tВведите Сумму: "; int temp; cin.get(); cin>>temp ; clients[searchID].Summ = temp; break; }**

**default: {cout << "\n\n\tВведено неверное значение, нажмите любую клавишу для возврата в меню... "; char p = \_getch(); system("cls"); break; }**

**}**

**system("cls");**

**}**

**Функция для внесения изменений данных клиента**

**void transformClients(person\* clients, int\* counter) {**

**std::cout << "\n\n\tИзменить данные клиента\n\n\t\tПоиск по \n\n\t1. ФИО\t2. Счёту в базе\t0. Выход\n\n\t";**

**int inputNum;**

**cin >> inputNum;**

**system("cls");**

**switch (inputNum) {**

**case(0): {cout << "\n\n\tВыход"; system("cls"); return; }**

**case(1): {**

**cout << "\n\n\tВведите Фамилию или Имя или Отчество. 0. Выход\n\n\t";**

**string inputStr;**

**cin >> inputStr;**

**system("cls");**

**if (inputStr == "0") { return; }**

**int searchID = -1;**

**int couErr = 0;**

**for (int i = 0; i < \*counter; i++) {**

**if (inputStr == clients[i].Surname) { searchID = i; couErr++; }**

**if (inputStr == clients[i].Name) { searchID = i; couErr++; }**

**if (inputStr == clients[i].Patronymic) { searchID = i; couErr++; }**

**}**

**if (couErr > 1) { cout << "\n\n\tНайдено более одного клиента, введите дргуие даннные клиента. Нажмите любую клавишу для возврата в меню... "; char p = \_getch(); system("cls"); return; }**

**if (searchID == -1) { cout << "\n\n\tКлиент не найден, нажмите любую клавишу что-бы вернуться в меню... "; char p = \_getch(); system("cls"); return; }**

**if (searchID >= 0) {**

**cout << "\n\n\tИзменить данные клиента " << clients[searchID].Surname << " " << clients[searchID].Name**

**<< " " << clients[searchID].Patronymic << "\n\n\t1. Да\t2. Нет\n\n\t";**

**int inputH;**

**cin >> inputH;**

**system("cls");**

**switch (inputH) {**

**case(1): {transform(clients, counter, searchID); return; }**

**case(2): { return; }**

**default: {cout << "\n\n\tВведено неверное значение, нажмите любую клавишу для возврата в меню... "; char p = \_getch(); system("cls"); return; }**

**}**

**}**

**}**

**case(2): {**

**cout << "\n\n\tВведите номер в базе. 0. Выход\n\n\t";**

**int inputInt;**

**cin >> inputInt;**

**if (inputInt == 0)return;**

**inputInt--;**

**system("cls");**

**cout << "\n\n\tИзменить данные клиента " << clients[inputInt].Surname << " " << clients[inputInt].Name**

**<< " " << clients[inputInt].Patronymic << "\n\n\t1. Да\t2. Нет\n\n\t";**

**int inP;**

**cin >> inP;**

**system("cls");**

**switch (inP) {**

**case(1): {transform(clients, counter, inputInt); return; }**

**case(2): { return; }**

**default: {cout << "\n\n\tВведено неверное значение, нажмите любую клавишу для возврата в меню... "; char p = \_getch(); system("cls"); return; }**

**}**

**}**

**default: {cout << "\n\n\tВведено неверное значение, нажмите любую клавишу для возврата в меню... "; char p = \_getch(); system("cls"); return; }**

**}**

**std::system("cls");**

**std::system("cls");**

**}**

**//finished**

**Подфункция удаления данных клиента**

**void swapAndDelClients(person\* clients, int\* counter, int searchID) {**

**for (int i = searchID; i < \*counter; i++) {**

**clients[i].Surname = clients[i + 1].Surname;**

**clients[i].Name = clients[i + 1].Name;**

**clients[i].Patronymic = clients[i + 1].Patronymic;**

**clients[i].Address = clients[i + 1].Address;**

**clients[i].PhoneNum = clients[i + 1].PhoneNum;**

**clients[i].PayDay = clients[i + 1].PayDay;**

**clients[i].Summ = clients[i + 1].Summ;**

**}**

**(\*counter)--;**

**}**

**Функция поиска и удаления клиента**

**void delClients(person\* clients, int\* counter) {**

**std::cout << "\n\tУдалить данные клиента\n\n\t\tПоиск по \n\n\t1. ФИО\t2. Счёту в базе\t0. Выход\n\n\t";**

**int inputNum;**

**cin >> inputNum;**

**system("cls");**

**switch (inputNum) {**

**case(0): {cout << "\n\n\tВыход"; system("cls"); return; }**

**case(1): {**

**cout << "\n\n\tВведите Фамилию или Имя или Отчество. 0. Выход\n\n\t";**

**string inputStr;**

**cin >> inputStr;**

**system("cls");**

**if (inputStr == "0") { return; }**

**int searchID = -1;**

**int couErr = 0;**

**for (int i = 0; i < \*counter; i++) {**

**if (inputStr == clients[i].Surname) { searchID = i; couErr++; }**

**if (inputStr == clients[i].Name) { searchID = i; couErr++; }**

**if (inputStr == clients[i].Patronymic) { searchID = i; couErr++; }**

**}**

**if (couErr > 1){cout << "\n\n\tНайдено более одного клиента, введите дргуие даннные клиента. Нажмите любую клавишу для возврата в меню... "; char p = \_getch(); system("cls"); return;}**

**if (searchID == -1) { cout << "\n\n\tКлиент не найден, нажмите любую клавишу что-бы вернуться в меню... "; char p = \_getch(); system("cls"); return; }**

**if (searchID >= 0) {**

**cout << "\n\n\tУдалить данные клиента " << clients[searchID].Surname << " " << clients[searchID].Name**

**<< " " << clients[searchID].Patronymic<<"\n\n\t1. Да\t2. Нет\n\n\t";**

**int inputH;**

**cin >> inputH;**

**system("cls");**

**switch (inputH) {**

**case(1): {swapAndDelClients(clients, counter, searchID);**

**cout << "\n\n\tУспешно удалено, нажмите любую клавишу для выхода в меню... "; char p = \_getch(); system("cls"); return;**

**}**

**case(2): { return; }**

**default: {cout << "\n\n\tВведено неверное значение, нажмите любую клавишу для возврата в меню... "; char p = \_getch(); system("cls"); return; }**

**}**

**}**

**}**

**case(2): {**

**cout << "\n\n\tВведите номер в базе. 0. Выход\n\n\t";**

**int inputInt;**

**cin >> inputInt;**

**if (inputInt == 0)return;**

**inputInt--;**

**system("cls");**

**cout << "\n\n\tУдалить данные клиента " << clients[inputInt].Surname << " " << clients[inputInt].Name**

**<< " " << clients[inputInt].Patronymic << "\n\n\t1. Да\t2. Нет\n\n\t";**

**int inP;**

**cin >> inP;**

**system("cls");**

**switch (inP) {**

**case(1): {swapAndDelClients(clients, counter, inputInt);**

**cout << "\n\n\tУспешно удалено, нажмите любую клавишу для выхода в меню... "; char p = \_getch(); system("cls"); return;**

**}**

**case(2): { return; }**

**default: {cout << "\n\n\tВведено неверное значение, нажмите любую клавишу для возврата в меню... "; char p = \_getch(); system("cls"); return; }**

**}**

**}**

**default: {cout << "\n\n\tВведено неверное значение, нажмите любую клавишу для возврата в меню... "; char p = \_getch(); system("cls"); return; }**

**}**

**std::system("cls");**

**}**

**//finished**

**Функция вывода всех данных клиентов в виде таблицы**

**void showClients(person\* clients, int\* counter) {**

**std::cout << "\n\n\tВывести список клиентов\n\n";**

**Вычисление размеров каждого столбца таблицы**

**int maxLenSurname = 7;**

**int maxLenName = 3;**

**int maxLenPatronymic = 8;**

**int maxLenAddress = 15;**

**int maxLenPhoneNum = 16;**

**int maxLenPayDay = 11;**

**int maxLenSumm = 13;**

**int LenSurname;**

**int LenName;**

**int LenPatronymic;**

**int LenAddress;**

**int LenPhoneNum;**

**int LenPayDay;**

**int LenSumm;**

**int maxSumm = 0;**

**for (int i = 0; i < \*counter; i++) {**

**LenSurname = clients[i].Surname .length();**

**LenName = clients[i].Name .length();**

**LenPatronymic = clients[i].Patronymic .length();**

**LenAddress = clients[i].Address .length();**

**LenPhoneNum = clients[i].PhoneNum .length();**

**LenPayDay = clients[i].PayDay .length();**

**LenSumm = 0;**

**int Summ = clients[i].Summ;**

**int g =0;**

**while (Summ>0) { Summ /= 10; LenSumm++; }**

**maxSumm += clients[i].Summ;**

**if (LenSurname > maxLenSurname) { maxLenSurname = LenSurname; }**

**if (LenName > maxLenName) { maxLenName = LenName; }**

**if (LenPatronymic > maxLenPatronymic) { maxLenPatronymic = LenPatronymic; }**

**if (LenAddress > maxLenAddress) { maxLenAddress = LenAddress; }**

**if (LenPhoneNum > maxLenPhoneNum) { maxLenPhoneNum = LenPhoneNum; }**

**if (LenPayDay > maxLenPayDay) { maxLenPayDay = LenPayDay; }**

**if (LenSumm > maxLenSumm) { maxLenSumm = LenSumm; }**

**}**

**int maxLength = maxLenSurname + maxLenName + maxLenPatronymic + maxLenAddress + maxLenPhoneNum + maxLenPayDay + maxLenSumm + 22;**

**Вывод таблицы**

**cout << "\t";**

**for (int i = 0; i < maxLength; i++) { cout << "-"; }**

**cout << endl;**

**printf\_s("\t| %\*s | %\*s | %\*s | %\*s | %\*s | %\*s | %\*s |",**

**maxLenSurname, "Фамилия",**

**maxLenName, "Имя",**

**maxLenPatronymic, "Отчество",**

**maxLenAddress, "Домашний адресс",**

**maxLenPhoneNum, "Телефонный номер",**

**maxLenPayDay, "День оплаты",**

**maxLenSumm, "Сумма покупки");**

**cout << "\n\t";**

**for (int i = 0; i < maxLength; i++) { cout << "-"; }**

**cout << "\n";**

**for (int i = 0; i < \*counter; i++) {**

**printf\_s(**

**"\t| %\*s | %\*s | %\*s | %\*s | %\*s | %\*s | %\*d |",**

**maxLenSurname, clients[i].Surname. c\_str(),**

**maxLenName, clients[i].Name. c\_str(),**

**maxLenPatronymic, clients[i].Patronymic. c\_str(),**

**maxLenAddress, clients[i].Address. c\_str(),**

**maxLenPhoneNum, clients[i].PhoneNum. c\_str(),**

**maxLenPayDay, clients[i].PayDay. c\_str(),**

**maxLenSumm, clients[i].Summ**

**);**

**cout << "\n\t";**

**for (int i = 0; i < maxLength; i++) { cout << "-"; }**

**cout << endl;**

**}**

**cout << "\t";**

**for (int i = 0; i < maxLength - (maxLenPayDay + 6); i++) { cout << " "; }**

**printf\_s("| %\*d |", maxLenSumm, maxSumm); cout << "\n\t";**

**for (int i = 0; i < maxLength - (maxLenPayDay + 6); i++) { cout << " "; }**

**for (int i = 0; i < maxLenPayDay + 6; i++) { cout << "-"; }**

**std::cout << "\n\tНажмите любую клавишу для возврата в меню... "; char p = \_getch(); std::system("cls");**

**std::system("cls");**

**}**

**//finished**

**Подфункция выводящая найденного клиента**

**void show(person\* clients, int\* counter, int searchID) {**

**cout << "\n\n\tФИО: " << clients[searchID].Surname << " " << clients[searchID].Name << " " << clients[searchID].Patronymic << " Адресс: "**

**<< clients[searchID].Address << "\n\tТелефоный номер: " << clients[searchID].PhoneNum << " День оплаты: " << clients[searchID].PayDay << " Сумма: " << clients[searchID].Summ;**

**}**

**Поиск клиента по ФИО**

**void searchClients(person\* clients, int\* counter) {**

**std::cout << "\n\tПоиск клиента по базе";**

**std::cout << "\n\n\tВведите Фамилию или Имя или Отчество\t0. Выход\n\n\t";**

**string inputStr;**

**cin >> inputStr;**

**system("cls");**

**if (inputStr == "0") { return; }**

**int searchID = -1;**

**for (int i = 0; i < \*counter; i++) {**

**if (inputStr == clients[i].Surname) { searchID = i; show(clients, counter, searchID); }**

**if (inputStr == clients[i].Name) { searchID = i; show(clients, counter, searchID); }**

**if (inputStr == clients[i].Patronymic) { searchID = i; show(clients, counter, searchID); }**

**}**

**if (searchID >= 0) { cout << "\n\n\tНажмите любую клавишу что-бы вернуться в меню... "; char p = \_getch(); system("cls"); return; }**

**if (searchID == -1) { cout << "\n\n\tКлиент не найден, нажмите любую клавишу что-бы вернуться в меню... "; char p = \_getch(); system("cls"); return; }**

**std::system("cls");**

**}**

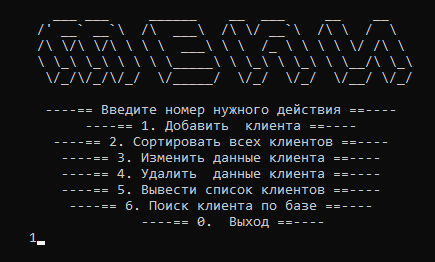
**//finish**

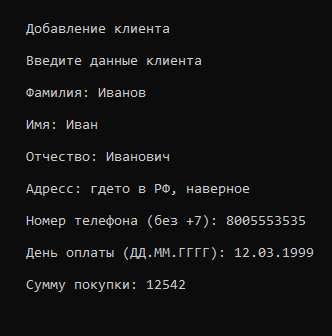
**#pragma endregion**

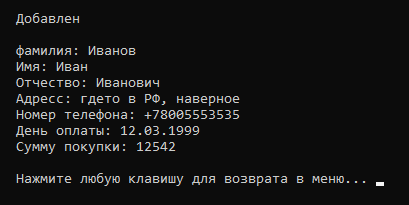
**3.4 Тестирование**

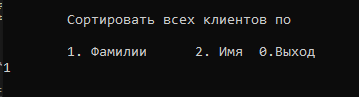
Работа в штатном режиме, введём валидные данные.



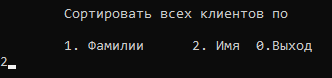




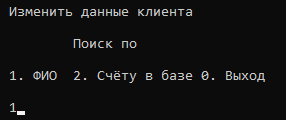


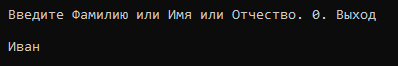


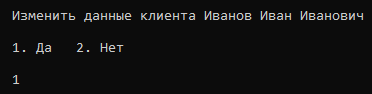


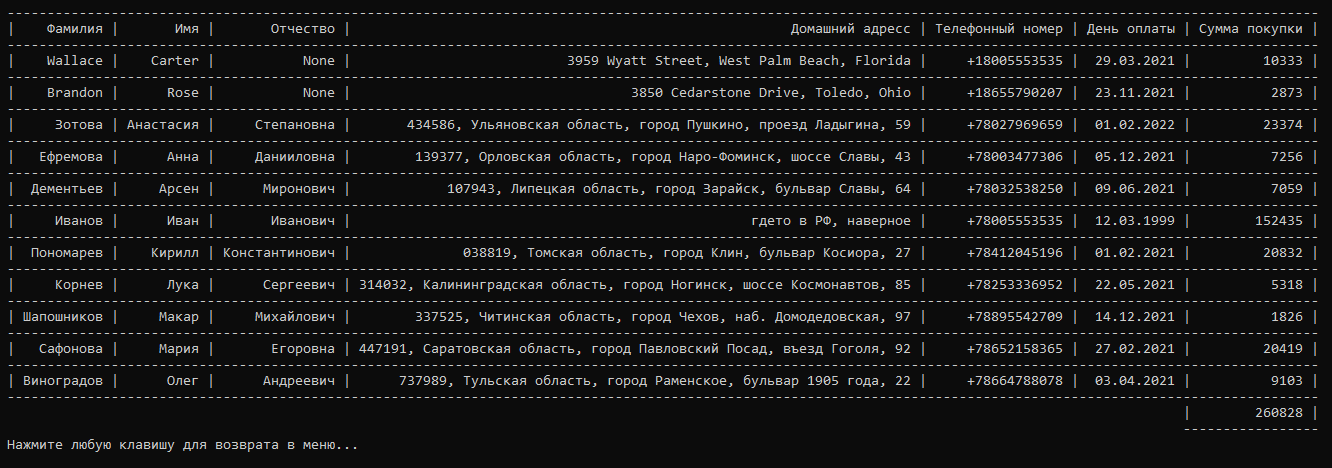
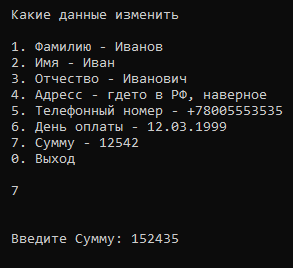


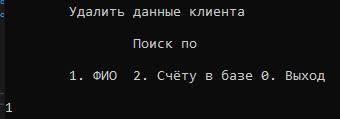




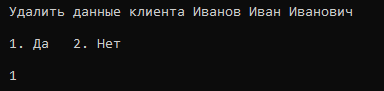


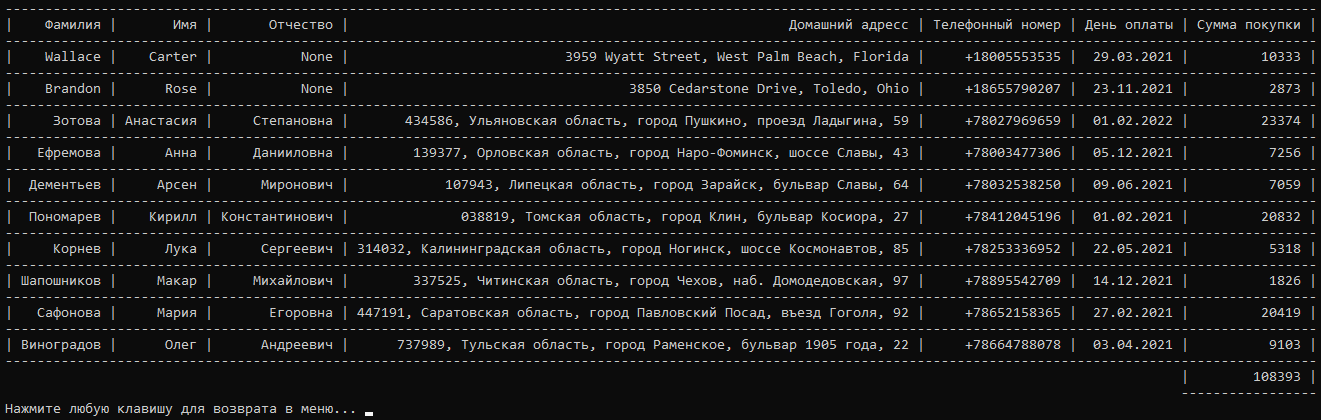


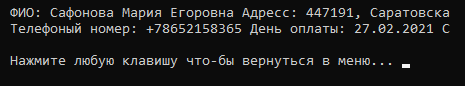
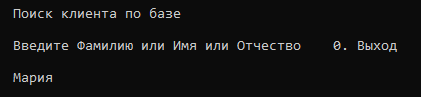












**Дополнительная задача**

**4 Словесная постановка задачи**

Разработать программу, создайте переменную структуру (структура с объединением в качестве элемента), представляющую собой универсальный массив, который может быть использован для записи целых чисел или строки. Внутри структуры создайте переменную-метку (флаг), которая говорит о том, какого типа данные сейчас в экземпляре структуры. Пользователь должен сам выбрать, что записать в экземпляр данной структуры. После записи выведите данные обратно на экран.

**4.1 Решение задачи**

**4.2 Формальная постановка задачи**

Входные данные:

MyStruct – структура, содержащее целочисленную переменную Type и объединение, которое содержит целочисленный и символьные массивы

SIZE - целое число, содержащее размер массивов в объединении;

Выходные данные:

**Директива препроцессора:** Эти данные объявляются так:

#define SIZE 10

**Структура:** Эти данные объявляются так:

struct MyStruct {

int Type;

union {…}TypeUnion;

}TypeStruct;

**Объединение:** Эти данные объявляются так:

union {

int x[SIZE];

char c[SIZE];

}TypeUnion;

**4.3 Реализация**

Перед реализацией additionalTask объявлена структура состоящая из целочисленной переменной и объединения, которое в свою очередь состоит из целочисленного и символьного массивов. В самой функции additionalTask вводится целочисленная переменная в структуру, а затем в зависимости введенного значения вводится числа или буквы, в массив в объединении. После выводится в виде строки

#pragma region additionalTask

#define SIZE 10

struct MyStruct {

int Type;

union {

int x[SIZE];

char c[SIZE];

}TypeUnion;

}TypeStruct;

void additionalTask() {

while (true) {

cout << "\n\n\tВведите тип, который будет записан в массив, в виде \n\n\t1. Чисел 2. Букв 0. Выход\n\n\t";

cin >> TypeStruct.Type;

system("cls");

switch (TypeStruct.Type) {

case(0):return;

case(1):{

cout << "\n\n\tВвод чисел\n\n";

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

cout << "\tВведите " << i + 1 << " число: ";

cin >> TypeStruct.TypeUnion.x[i];

}

system("cls");

cout << "\n\n\tВы ввели: ";

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

cout << TypeStruct.TypeUnion.x[i]<<” “;

}

cout << "\n\n\tДля выхода, нажмите любую клавишу... "; char p = \_getch(); system("cls");

return;

}

case(2):{

cout << "\n\n\tВвод букв\n";

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

cout << "\n\tВведите " << i + 1 << " букву: ";

cin >> TypeStruct.TypeUnion.c[i];

}

system("cls");

cout << "\n\n\tВы ввели: ";

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

cout << TypeStruct.TypeUnion.c[i];

}

cout << "\n\n\tДля выхода, нажмите любую клавишу... "; char p = \_getch(); system("cls");

return;

}

default:{cout << "\n\n\tВведено неверное значение, нажмите любую клавишу, чтобы повторно ввести тип... "; char p = \_getch(); system("cls"); continue;}

}

if (TypeStruct.Type > 2 || TypeStruct.Type < 0) {cout << "\n\n\tВведено неверное значение, нажмите любую клавишу, чтобы повторно ввести тип... "; char p = \_getch(); system("cls"); continue;}

if (TypeStruct.Type == 0) {return;}

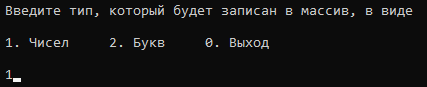
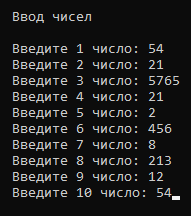
}

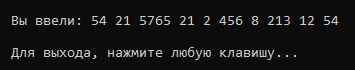
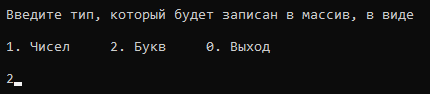
}

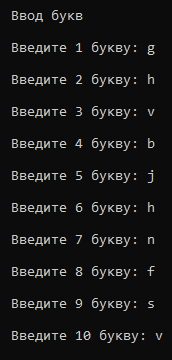
#pragma endregion

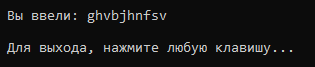
**4.4 Тестирование**

Работа в штатном режиме, введём валидные данные





**Заключение**

В этой лабораторной работе я научился работать со структурами, объединениями, массивами структур, битовыми полями и пользовательскими типами данных.

