Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

ЛАБОРАТОРНАЯ РАБОТА

ПО ОСНОВАМ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЮ

За 1 семестр

Вариант № 9

Выполнила:

студента группы РИС-20-1бзу

Канзепаров Р. Р.

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

Пермь,

2021

**Лабораторная работа № 7**

**7.1. «Перегрузка функций в Си++»**

**Цель:** Знакомство с организацией перегруженных функций в Си++.

**1 Постановка задачи**

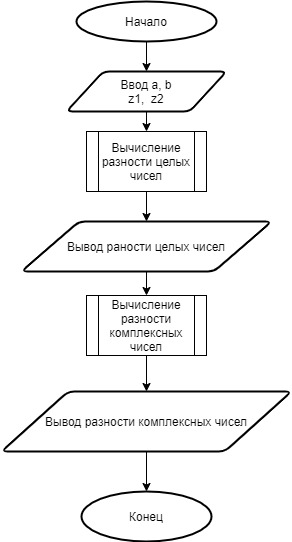
Написать перегруженные функции и основную программу, которая их вызывает.

**2 Задание варианта**

а) для сложения десятичных дробей;

б) для сложения обыкновенных дробей.

**3 Блок-схема решения**



Ввод a,b,e,d

Вывод суммы обыкновенных дробей

Сумма обык. дробей

Вывод суммы десятичных дробей

Сумма десятичных дробейsf

**4 Листинг программы**

#include <cstdlib>

#include <iostream>

using namespace std;

float sum(float a, float b)

{

return a + b;

}

double sum(double a, double b)

{

return a + b;

}

int main(int argc, char\* argv[])

{

float a, b;

double e, e1, d, d1;

cout << "int a= ";

cin >> a;

cout << "int b= ";

cin >> b;

cout << "razn " << sum (a, b) << endl;

cout << "double e= ";

cin >> e;

cout << "double e1= ";

cin >> e1;

cout << "double d= ";

cin >> d;

cout << "double d1= ";

cin >> d1;

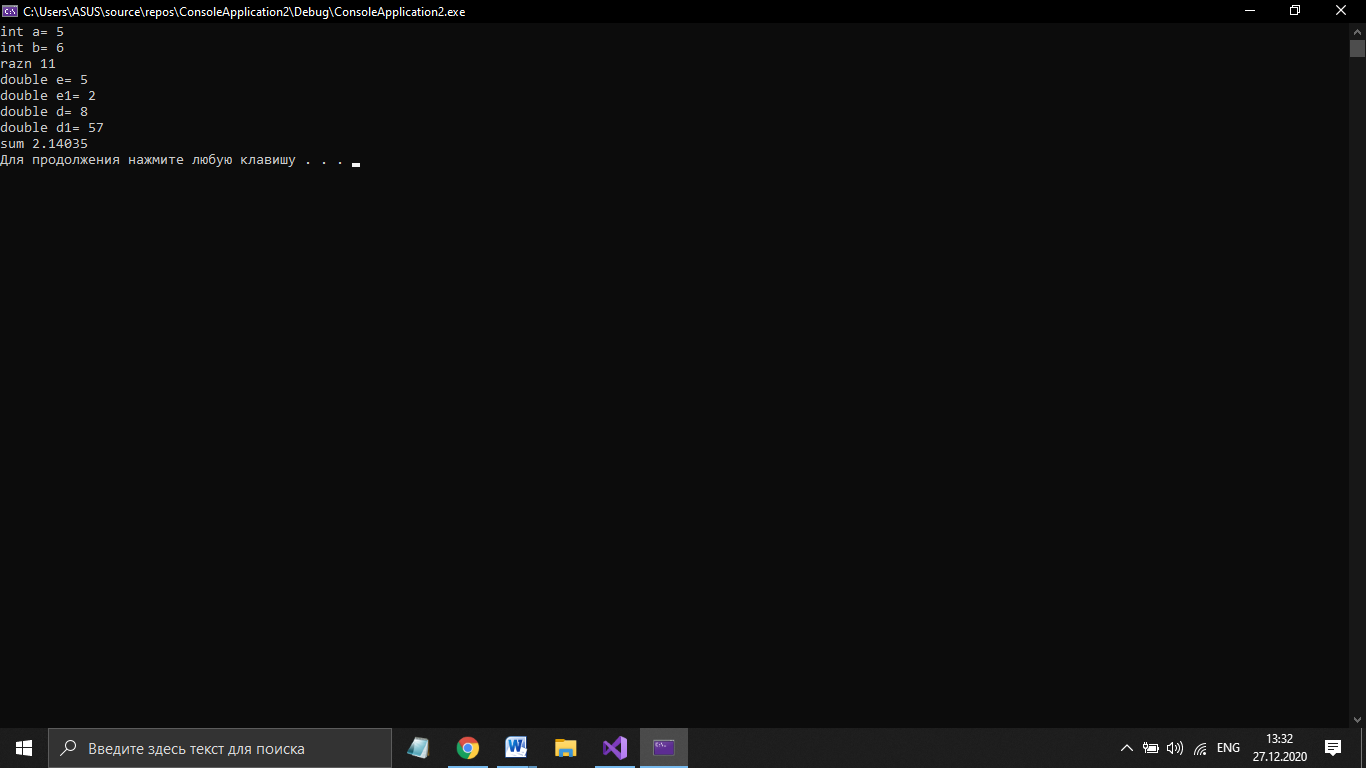
cout << "sum " << sum((e, e1), (d / d1)) << endl;

system("PAUSE");

return EXIT\_SUCCESS;

}

**5 Результаты работы программы**



**7.2. «Функции с переменным числом параметров»**

**Цель:** Знакомство с организацией функций с переменным числом параметров.

**1 Постановка задачи**

Решить указанную в варианте задачу, используя функции с переменным числом параметров.

**2 Задание варианта**

Написать функцию sum с переменным числом параметров, которая находит сумму заданных обыкновенных дробей. Написать вызывающую функцию main, которая обращается к функции sum не менее трех раз с количеством параметров 5, 10, 12.

**3 Блок-схема решения**

**4 Листинг программы**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include<Windows.h>

//Включение макросов

#include <stdarg.h>

using namespace std;

//Объявление функции, вычисляющей сумму чисел типа int по формуле:S = a1 \* a2 + a3 \* a4 + a5 \* a6 + ....

double sum(double a, ...);

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << "Сумма пяти чисел : " << sum(5, 1.5, 2.3, 3.2, 4.7, 5) << endl;

cout << "Сумма десяти чисел : " << sum(10,0.1, 0.6, 3.2, 4.8, 5.6, 6.1, 7.3, 8.2, 9.4, 10) << endl;

cout << "Сумма двенадцати чисел : " << sum(12, 1.1, 2.3, 3.9, 4.03, 5.5, 6.1, 7.2, 8.1, 9.3, 10.1, 11.2, 12) << endl;

return 0;

}

double sum(double a, ...)

{

va\_list arg;

//Инициализация указателя

va\_start(arg, a);

double s = 0;

for (double i = 0; i < a; i += 2) {

//Возвращение значения очередного аргумента и присваивание его переменной

int x = va\_arg(arg, int);

int y = va\_arg(arg, int);

s += x \* y;

}

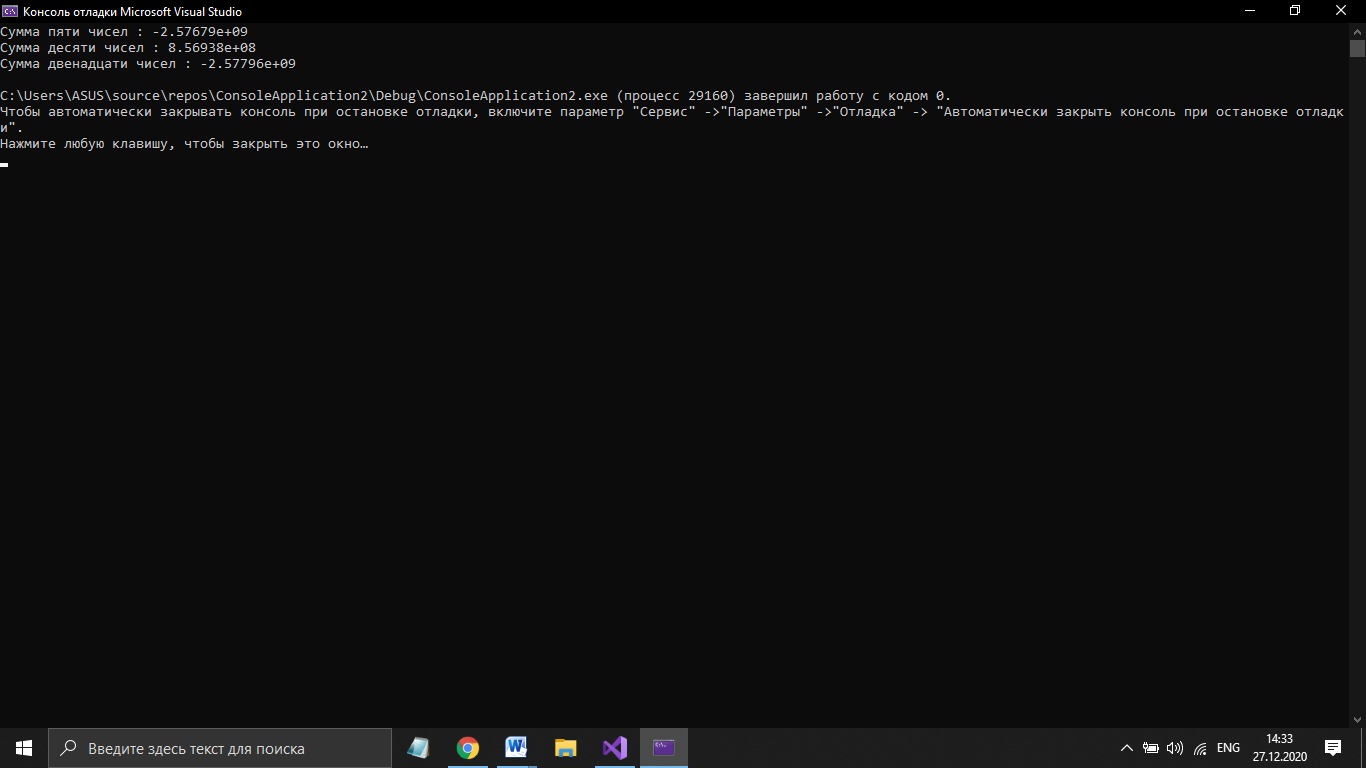
//Завершение перебора аргументов

va\_end(arg);

return s;

}

**5 Результаты работы программы**



**Лабораторная работа №8**

**«Блоковый ввод-вывод»**

**Цель:** Работа с двоичными файлами, организация ввода-вывода структурированной информации и ее хранение на внешних носителях.

**1 Постановка задачи**

Сформировать двоичный файл из элементов, заданной в варианте структуры, распечатать его содержимое, выполнить удаление и добавление элементов в соответствии со своим вариантом, используя для поиска удаляемых или добавляемых элементов функцию. Формирование, печать, добавление и удаление элементов оформить в виде функций. Предусмотреть сообщения об ошибках при открытии файла и выполнении операций ввода/вывода.

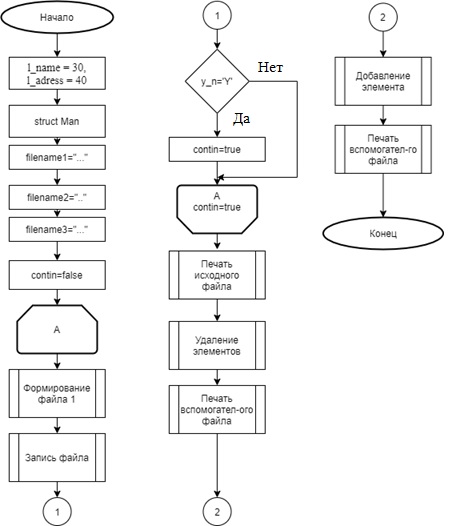
**2 Задание варианта**

Структура "Пациент":

* фамилия, имя, отчество;
* домашний адрес;
* номер медицинской карты;
* номер страхового полиса.

Удалить элемент с заданным номером медицинской карты, добавить 2 элемента в начало файла.

**3 Блок – схема решения**



**4 Листинг программы**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include<Windows.h>

#include <stdio.h>

#include <stdlib.h>

// Длина поля под ФИО и адрес

const int l\_name = 50, l\_adress = 50;

//Объявление структуры

struct Pacient {

char name[l\_name + 1];

char adress[l\_adress + 1];

int medkart;

int strahpolis;

};

using namespace std;

int zapis(FILE\* f);

int formirovanie(const char\* filename1);

int print(const char\* filename);

int del(const char\* filename, const char\* filename2);

int dob(const char\* filename, const char\* filename1);

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

bool contin;

char y\_n[2];

//Исходный файл

const char filename1[] = "g:\\text1.bin";

//Вспомогательный файл для удаления

const char filename2[] = "g:\\text11.bin";

//Вспомогательный файл для добавления

const char filename3[] = "g:\\text.bin";

//Цикл записи файла

do {

contin = false;

if (formirovanie(filename1) != 0) {

puts("Ошибка при записи в файл");

return 1;

}

cout << endl;

puts("Продолжить (y/n)?");

gets\_s(y\_n);

//Проверка записаны ли файл данные

if ((y\_n[0] == 'y') || y\_n[0] == 'Y') contin = true;

} while (contin);

//Вызов функции печати исходного файла

print(filename1);

cout << endl;

//Вызов функции удаления с помощью вспомогательного файла

del(filename1, filename2);

cout << endl;

//Вызов функции добавления с помощью вспомогательного файла

dob(filename2, filename3);

cout << endl;

return 0;

}

//Определение функции, формирование исходного файла

int formirovanie(const char\* filename1) {

FILE\* fout;

if ((fout = fopen(filename1, "ab")) == NULL) {

puts("Ошибка открытия вых. файла\n");

}

//Вызов функции записи структуры в исходный файл

zapis(fout);

fclose(fout);

return 0;

}

//Определение функции, запись структуры

int zapis(FILE\* f) {

//Определение структуры

Pacient pacient;

char buf[80];

char name[l\_name + 1];

char adress[l\_adress + 1];

puts("Введите ФИО:");

gets\_s(name);

gets\_s(name);

if (strlen(name) < l\_name)

for (int i = strlen(name); i < l\_name; i++)

name[i] = ' ';

name[l\_name] = 0;

//Копирование введённого имени в структуру с длиной не более l\_name+1

strncpy(pacient.name, name, l\_name + 1);

puts("Введите адрес: "); gets\_s(adress);

if (strlen(adress) < l\_adress)

for (int i = strlen(adress); i < l\_adress; i++)

adress[i] = ' ';

adress[l\_adress] = 0;

strncpy(pacient.adress, adress, l\_adress + 1);

do {

puts("Введите номер мед.карты: "); gets\_s(buf);

//Перевод строки в длинное целое число

} while ((pacient.medkart = atol(buf)) == 0);

do {

puts("Введите номер страх.полис: "); gets\_s(buf);

//Перевод строки в целое число

} while ((pacient.strahpolis = atoi(buf)) == 0);

//Занесение структуры в файл

int str = fwrite(&pacient, sizeof(pacient), 1, f);

if (str == 1) return 0;

else return 2;

}

//Определение функции, печать файла

int print(const char\* filename) {

Pacient pacient;

//Указатель на файл

FILE\* fout;

if ((fout = fopen(filename, "rb")) == NULL) {

puts("Ошибка открытия файла\n");

return 1;

}

printf("\n");

printf("Содержание файла:\n");

//Установка позиции с конца

fseek(fout, 0, SEEK\_END);

// Значение длины файла в записях

int n\_rec = ftell(fout) / sizeof(pacient);

int i = 0;

while (true) {

if (i < 0 || i >= n\_rec) break;

//Установка позиции с начала

fseek(fout, i \* sizeof(pacient), SEEK\_SET);

//Чтение построчно из файла и записывание в man

fread(&pacient, sizeof(pacient), 1, fout);

printf("%30s%10s%17i%5i\n", pacient.name, pacient.adress, pacient.medkart, pacient.strahpolis);

i++;

}

return 0;

}

int del(const char\* filename, const char\* filename2) {

int medkart;

Pacient pacient;

FILE\* f;

//Открытие исходного файла для чтения в двоичном режиме

if ((f = fopen(filename, "rb")) == NULL) {

puts("Ошибка открытия исходного файла\n");

return 1;

}

FILE\* temp;

//Открытие вспомогательного файла для записи в двоичном режиме

if ((temp = fopen(filename2, "wb")) == NULL) {

puts("Ошибка открытия вспомогательного файла\n");

return 1;

}

printf("Введите номер мед.карты:\n");

scanf("%i", &medkart);

//Выполняется чтение пока не достигнут конец файла

while (fread(&pacient, sizeof(pacient), 1, f)) {

//Проверка совпадает ли введённый возраст с возрастом структуры

if (pacient.medkart != medkart) {

//Если нет, записываем значение в вспомогательный файл

fwrite(&pacient, sizeof(pacient), 1, temp);

}

//Если да, то не записываем в вспомогательный файл

else printf("Удалено\n");

}

fclose(f);

fclose(temp);

//Вызов функции печати вспомогательного файла

print(filename2);

return 0;

}

//Определение функции, добавление нового элемента

int dob(const char\* filename, const char\* filename1) {

//Номер записи

int num;

Pacient pacient;

FILE\* f;

//Открытие 1-го вспомогательного файла для чтения в двоичном режиме

if ((f = fopen(filename, "rb")) == NULL) {

puts("Ошибка открытия исходного файла\n");

return 1;

}

FILE\* temp;

//Открытие 2-го вспомогательного файла для чтения в двоичном режиме

if ((temp = fopen(filename1, "ab")) == NULL) {

puts("Ошибка открытия вспомогательного файла\n");

}

int i = 0;

printf("Введите номер элемента после которого необходимо добавить элемент:\n");

scanf("%i", &num);

//Выполняется чтение пока не достигнут конец файла

while (fread(&pacient, sizeof(pacient), 1, f)) {

//Запись, если введённый номер не совпадает с номером записи…

fwrite(&pacient, sizeof(pacient), 1, temp);

//если совпадает, вызывается функция записи нового элемента

if (num == i)

zapis(temp);

i++;

}

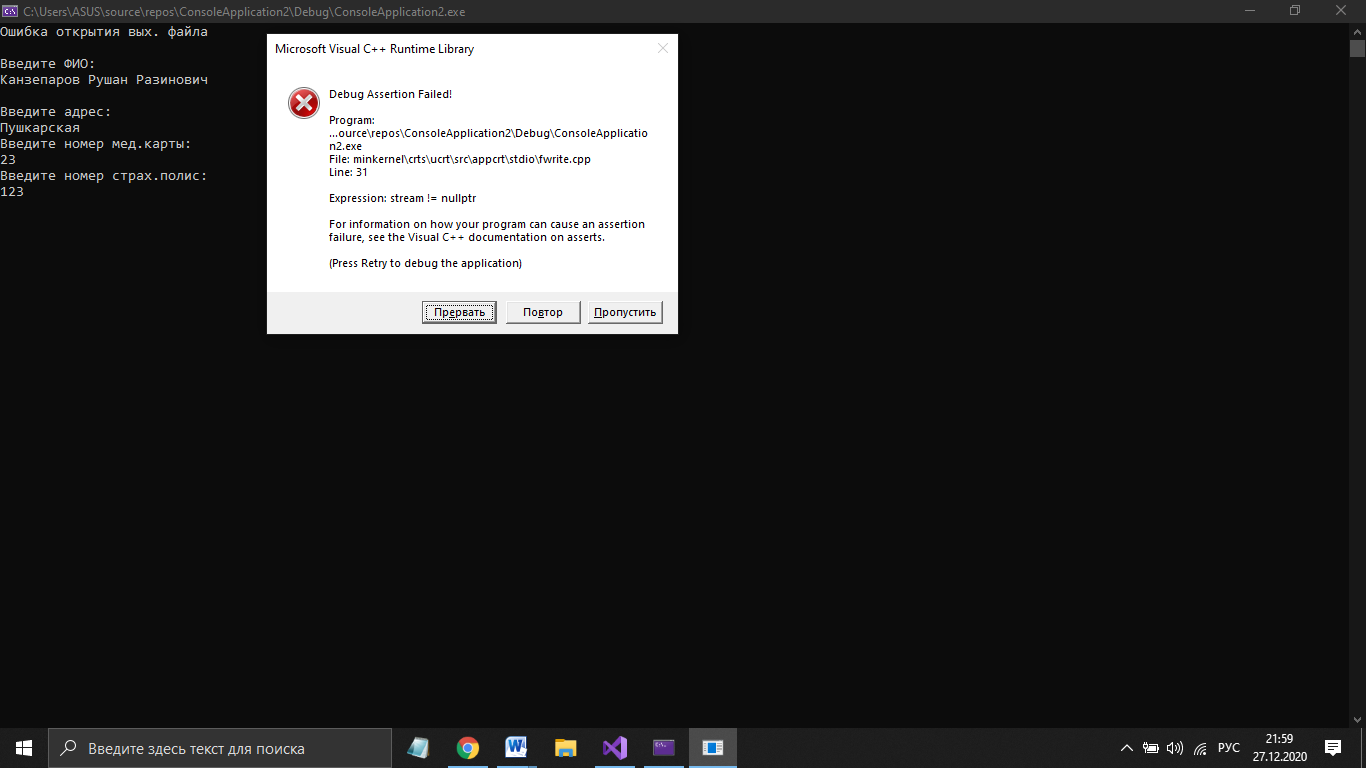
fclose(f);

fclose(temp);

print(filename1);

return 0;

}**5 Результаты работы программы**



**Лабораторная работа № 9**

**«Строковый ввод-вывод»**

**Цель:** Работа с текстовыми файлами, ввод-вывод текстовой информации и ее хранение на внешних носителях.

**1 Постановка задачи**

1. Создать текстовый файл F1 не менее, чем из 10 строк и записать в него информацию
2. Выполнить задание.

**2 Задание варианта**

1. Скопировать из файла F1 в файл F2 все строки, которые содержат только одно слово.
2. Найти самое длинное слово в файле F2.

**3 Блок – схема решения**

**4 Листинг программы**

} #include<iostream>

#include<sstream>

#include<fstream>

#include<string>

using namespace std;

int main(){int i; string s,sm,ins,outs;

setlocale(LC\_ALL, "Russian");

cout<<"Имя входного файла: ";

cin>>ins; ifstream ifs(ins);if(ifs.fail())

{cerr<<"Ошибка открытия входного файла "<<endl;

cin.get();

return 1;

}cout<<"Name of the output file: ";

cin>>outs; ofstream ofs(outs);

cout<<endl;while(getline(ifs,s))

{istringstream ist(s);

for(i=0;ist>>sm;i++);

if(i==1)ofs<<s<<'\n';

}ifs.close();

ofs.close(); ifstream kfs(outs);

sm.clear();

while(kfs>>s)

if(s.length()>sm.length())sm=s;

kfs.close();

cout<<sm<<endl;

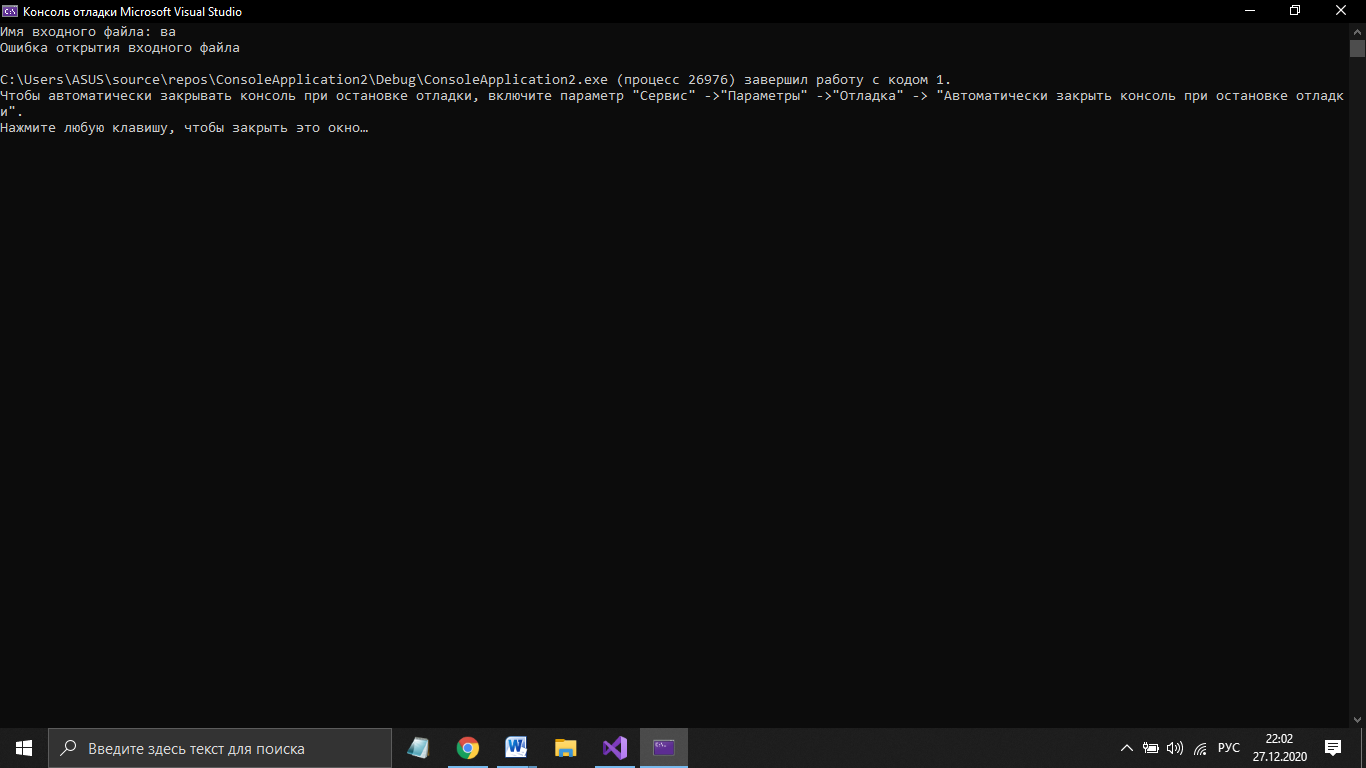
cout<<"\nPress Enter to exit\n";

cin.get();

cin.get();

}

**5 Результаты работы программы**



**Лабораторная работа № 10**

**«Динамические массивы»**

**Цель:** Организация динамических массивов.

**1 Постановка задачи**

Написать программу, в которой создаются динамические массивы и выполнить их обработку в соответствии со своим вариантом.

**2 Задание варианта**

Сформировать двумерный массив. Добавить в него столбец с заданным номером;

**3 Блок – схема решения**

.

**4 Листинг программы**

#include <iostream>

#include <time.h>

#include <cstdlib>

using namespace std;

int main()

{

srand(time(NULL));

int i, j, n, m, k;

int a[100][100], K[100];

setlocale(0, "");

cout << "Введите количество строк:";

cin >> n;

cout << "Введите количество столбцов:";

cin >> m;

for (i = 0; i < n; i++)

for (j = 0; j < m; j++)

a[i][j] = rand() % 100;

cout << "Матрица:" << endl;

for (i = 0; i < n; i++)

{

for (j = 0; j < m; j++)

cout << "\t" << a[i][j] << " ";

cout << endl;

}

cout << "Введите номер столбца, который необходимо вставить:"; cin >> k;

if (k > m + 1) cout << "Некорректное значение k";

else {

cout << "Введите элементы столбца:" << endl;

for (i = 0; i < n; i++) {

cin >> K[i];

}

cout << "Новая матрица:" << endl;

for (i = 0; i < n; i++)

{

for (j = 0; j < m; j++)

{

if (j == k - 1) {

cout << "\t" << K[i] << " " << "\t" << a[i][j] << " ";

}

else cout << "\t" << a[i][j] << " ";

}

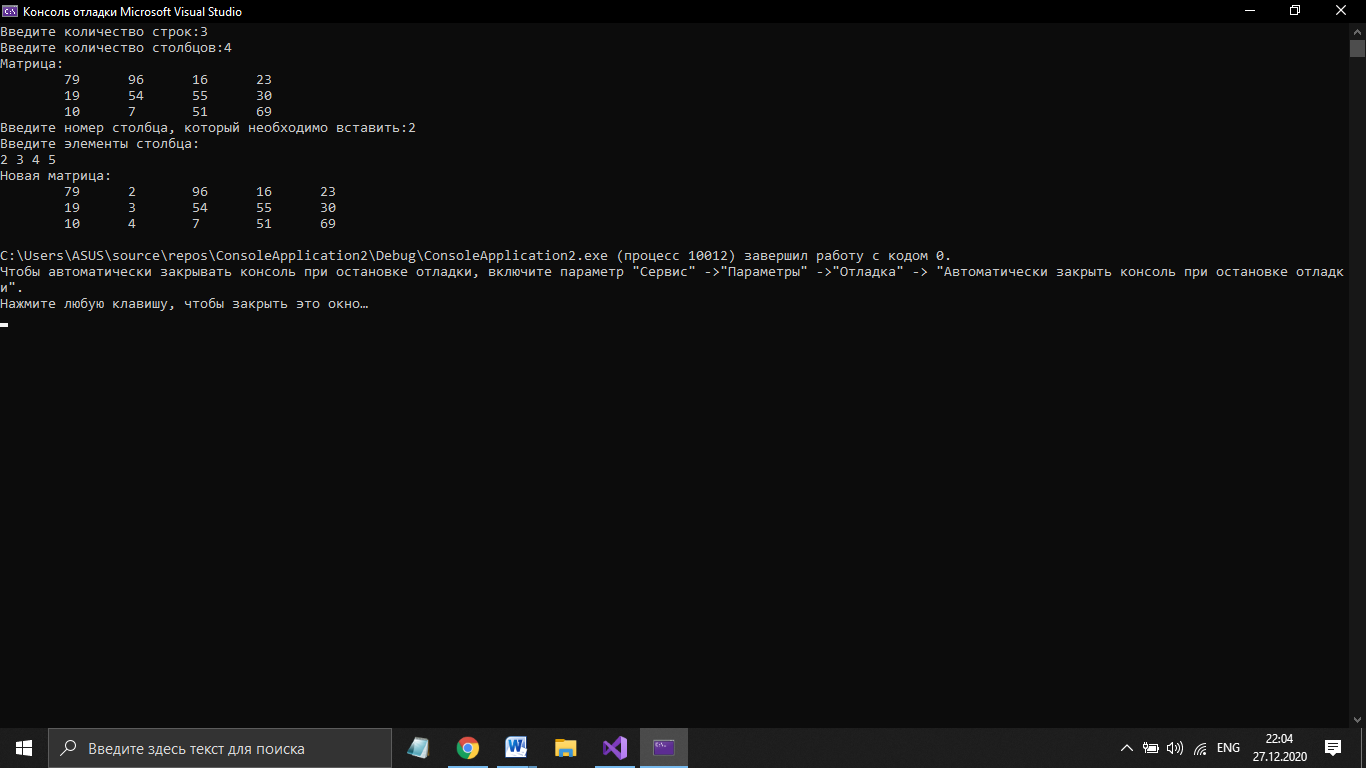
cout << endl;

}

}

return 0;

}**5 Результаты работы**



**Лабораторная работа № 11**

**«Информационные динамические структуры»**

**Цель:** Знакомство с динамическими информационными структурами на примере одно- и двунаправленных списков.

**1 Постановка задачи**

Написать программу, в которой создаются динамические структуры и выполнить их обработку в соответствии со своим вариантом.

**Для каждого вариант разработать следующие функции:**

1. Создание списка.
2. Добавление элемента в список (в соответствии со своим вариантом).
3. Удаление элемента из списка (в соответствии со своим вариантом).
4. Печать списка.
5. Запись списка в файл.
6. Уничтожение списка.
7. Восстановление списка из файла.

**2 Задание варианта**

Записи в линейном списке содержат ключевое поле типа int. Сформировать двунаправленный список. Удалить из него К элементов перед элементом с заданным номером, добавить К элементов в конец списка.

**3 Блок – схема решения**

.

**4 Листинг программы**

#define \_CRT\_SECURE\_NO\_WARNINGS

struct list {

struct list\* next;

int info;

};

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <stdlib.h>

#include <windows.h>

#include <clocale>

using namespace std;

/\* INPUT\_info() - ввод с клавиатуры информации \*/

int input\_info(struct list\* ptr) {

fflush(stdin);

scanf("%d", &(ptr->info));

return ptr->info;

}

/\* PRINTON() - Вывод информации на экран. Возвращает

указатель на следующий элемент списка. \*/

struct list\* printOn(struct list\* ptr) {

printf("%d ", ptr->info);

return (ptr->next);

}

/\* DISPLAY() - Выводит информацию во всех элементах списка.

Сканирует список от начала в конец. Возвращает указатель на

последний элемент. \*/

struct list\* display(struct list\* head) {

struct list\* ret;

if (head == NULL) { /\* если список пуст \*/

puts("Список пуст");

return NULL;

}

do {

printOn(head);

ret = head;

head = head->next;

} while (head != NULL);

return (ret);

}

/\* DESTROY() - Освобождает память, выделенную под элемент списка. \*/

void destroy(struct list\* element) {

free(element);

}

/\* HASMEMBER() - Проверяет, есть ли в списке, элемент с

идентичными полями. Список сканируется от начала в конец. \*/

int hasMember(struct list\* head, struct list\* work) {

while (head != NULL) { /\* цикл сканирования списка \*/

if (head->info == work->info) return 1;

head = head->next;

}

return(0);

}

/\* ADD() - Помещает элемент new\_ptr в список head в порядке

возрастания чисел (поля info). \*/

int add(struct list\*\* head, struct list\* new\_ptr) {

struct list\* first, \* second;

if ((\*head) == NULL) { /\* если список пуст \*/

(\*head) = new\_ptr;

new\_ptr->next = NULL;

return 0;

}

if ((\*head)->next == NULL) { /\* если в нем всего один элемент \*/

if ((\*head)->info > new\_ptr->info) {

/\* новый элемент становится первым в списке \*/

second = (\*head); /\* сохраним указатель на второй \*/

(\*head) = new\_ptr;

new\_ptr->next = second;

second->next = NULL;

}

else { /\* новый элемент становится в конец списка \*/

(\*head)->next = new\_ptr;

new\_ptr->next = NULL;

}

return 1;

}

else {

if ((\*head)->info > new\_ptr->info) {

/\* новый эл-т ставится первым в списке \*/

second = (\*head); /\* сохраним указатель на второй \*/

(\*head) = new\_ptr;

new\_ptr->next = second;

return 4;

}

first = (\*head);

second = first->next;

while (first->next != NULL) { /\* цикл поиска места в списке \*/

if (first->info <= new\_ptr->info &&

second->info >= new\_ptr->info) {

/\* вставляем элемент между first и second \*/

first->next = new\_ptr;

new\_ptr->next = second;

return 2;

}

first = second;

second = first->next;

}

/\* если добрались сюда, элемент ставим в конец списка \*/

first->next = new\_ptr;

new\_ptr->next = NULL;

return 3;

}

}

/\* DETACH() - Удаляет элемент element из списка head \*/

void detach(struct list\*\* head, struct list\* element) {

struct list\* prev;

if (\*head == NULL) return; /\* если список пуст \*/

if ((\*head) == element) { /\* если удаляемый эл-т - первый \*/

(\*head) = element->next;

destroy(element);

return;

}

/\* Цикл поиска предыдущего для element элемента. \*/

prev = (\*head);

while ((prev->next) != element)

prev = prev->next; /\* на выходе из цикла - адрес предыдущ\*/

prev->next = element->next;

destroy(element);

return;

}

int main(void) {

setlocale(LC\_ALL, "Russian");

char ch;

struct list\* head; /\* указатель на начало списка \*/

struct list\* new\_ptr;/\* указатель на новый элемент списка \*/

struct list\* cur; /\* указатель на текущий элемент списка \*/

struct list work;

head = NULL; /\* вначале список пуст \*/

puts("Введите числа по одному в одну строку, 0 - конец списка");

while (input\_info(&work) != 0) { /\* цикл ввода описаний \*/

if (hasMember(head, &work) != 1) {

if ((new\_ptr = (struct list\*)malloc(sizeof(struct list))) == NULL) {

puts("Нет памяти!");

break;

}

new\_ptr->info = work.info;/\* копируем введен. информацию \*/

add(&head, new\_ptr); /\* добавляем элемент в список \*/

}

}

if (head == NULL) /\* список остался пустым ? \*/

return 1; /\* да, завершение программы \*/

puts("\*\*\* ALL LIST \*\*\*");

display(head); /\* вывод всех элем-тов списка \*/

/\* Цикл выборочного удаления элементов, начиная с первого. \*/

cur = head;

puts("\*\*\* DELETING \*\*\*");

do {

new\_ptr = printOn(cur); /\* вывод элемента списка \*/

printf("Удалить строку? ( y/n) ");

if ((ch = \_getch()) == 'Y' || ch == 'y') {

puts("Да");

detach(&head, cur); /\* удаление элемента из спсика \*/

}

else puts("Нет");

cur = new\_ptr;

} while (new\_ptr != NULL);

puts("\*\*\* ALL LIST AFTER DELETING \*\*\*");

display(head); /\* вывод всех элем-тов списка \*/

system("Пауза");

return 0;

}

**5 Результаты работы программы**

