Министерство образования и науки Российской Федерации

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

федеральное государственное бюджетное образовательное учреждение

высшего образования

"Санкт-Петербургский государственный технологический институт

(технический университет)"

Кафедра САПР и У

Факультетинформатики и вычислительной техники

Курс 3

Группа 4291

Учебная дисциплина: «Разработка программных систем»

КОНТРОЛЬНАЯ РАБОТА №3

Вариант 3

Студент Борисова Д.С. Личная подпись Расшифровка подписи

Руководитель Личная подпись Расшифровка подписи должность

Г. Санкт-Петербург

2025 г.

**Контрольная работа №3**

Вариант 3

**Задание:**

Разработать программу, состоящую из нескольких модулей, которая в массиве чисел из Size элементов находит такую непрерывную последовательность элементов (1<M<Size-1), которая обладает свойством, соответствующему варианту и выводит индекс начального и индекс конечного элемента найденной последовательности. Программа должна предоставлять пользователю опциональный выбор критерия поиска, консольный ввод массива, отображение текущего массива, загрузку массива из файла, сохранение массива в файле, создание тестовых массивов, и диагностику ошибок во всех функциях программы.

Свойства массива:

Непрерывная последовательность чисел является пропорциональной либо на увеличение, либо на уменьшение (должно выбираться опционально).

Последовательность пропорциональна, если следующий элемент получается из предыдущего умножением на один и тот же коэффициент, например:

2,-14,-28,4-16,8-33,6 : (\*2)

**Математическая модель:**

Математическая модель для задачи "непрерывная последовательность чисел является пропорциональной либо на увеличение, либо на уменьшение" предполагает, что каждый элемент последовательности получается умножением предыдущего на постоянный коэффициент, называемый коэффициентом пропорциональности.

Последовательность чисел пропорциональна, если каждый следующий элемент получается умножением предыдущего на постоянный коэффициент rrr:

an+1=an⋅r для всех n≥1

Если r >1 r > 1 r >1, последовательность увеличивается.

Если 0< r < 10 < r < 10< r < 1, последовательность уменьшается.

Если r<0r < 0r<0, элементы чередуются по знаку.

Примеры:

* r=2r = 2r=2: 2,4,8,16,…
* r=0.5r =0.5: 2,1,0.5,0.25,…
* r=−2r = -2: 2,−4,8,−16,…

Алгоритм:

Для первого элемента a1​ и коэффициента r, каждый следующий элемент вычисляется как an+1=an⋅r.

**Техническое задание:**

Разработать программу поиска B массиве вещественных чисел непрерывной последовательности, удовлетворяющей критерию: максимум или минимум суммы элементов.

Требования к функциональности:

* консольный ввод-вывод массива чисел,
* файловый ввод-вывод массива чисел,
* обработка массива чисел для поиска последовательности,
* диагностика ошибок.

Требования к интерфейсу:

* диалоговое консольное меню ко всем функциям,
* наличие подсказок при при вводе данных и выборе команд,
* запрос подтверждений от пользователя в неоднозначных ситуациях,
* вывод понятной информации об ошибках.

Требования к тестированию:

* все функции программы должны иметь тестовые планы,
* программа должна хранить готовые тестовые массивы,
* отработать тестовые случаи при работе с файлами.

Требования к форматам хранения данных:

* разработать формат хранения массива в файле,
* составить спецификацию хранения данных.

Требования к архитектуре:

Программа должна строиться из четырёх модулей, каждый из которых отвечает за определенную функциональность программы, составить спецификацию на содержание и назначение модулей.

**Декомпозиция задач:**

Главная функция **main:**

1. Устанавливает локаль для корректной работы с русскими символами с помощью std::setlocale и std::wcout.imbue.
2. Инициализирует указатель на массив myArray как NULL и задает переменную arrSize для хранения размера массива.
3. Вызывает функцию MainMenu, передавая в неё указатель на массив и размер массива для дальнейшей обработки.

Основная цель — подготовить программу для работы с русскими символами и инициализировать массив перед запуском главного меню.

Функция **MainMenu** реализует консольное меню с интерфейсом пользователя, позволяя выполнять следующие действия:

1. Вводить новый массив.
2. Создавать тестовый массив.
3. Сохранять массив в файл.
4. Загружать массив из файла.
5. Отображать текущий массив.
6. Выполнять поиск в массиве по критериям (максимум или минимум суммы).
7. Завершить программу.

Меню работает в цикле до выбора команды выхода (Q).

Функция **IsRewrite** запрашивает у пользователя, хочет ли он перезаписать существующий файл. Если пользователь вводит 'Y' или 'y', функция возвращает true, в противном случае — false.

**TestArray**: Эта функция загружает тестовый массив в текущий массив. Если массив уже существует, он удаляется. Затем функция создает новый массив и копирует в него элементы из статического тестового массива.

**CalcCriteria**: Функция вычисляет сумму элементов массива от индекса left до индекса right (включительно).

**Search**: Функция ищет непрерывную последовательность элементов в массиве, которая имеет максимальную или минимальную сумму в зависимости от значения параметра opt. Параметры begin и end указывают на индексы начала и конца найденной последовательности.

**ShowArray**: Эта функция выводит на консоль элементы массива pArray размером Size. Если массив не инициализирован или его размер меньше 1, выводится сообщение "Массив не введен".

**InputArray**: Функция для ввода массива с консоли. Пользователь вводит коэффициент для увеличения или уменьшения последовательности. Функция проверяет корректность размера массива (от 1 до 1000 элементов), затем заполняет его значениями, начиная с первого элемента и продолжая с учетом выбранного коэффициента для увеличения или уменьшения значений.

**Error**: Эта функция выводит описание ошибки в зависимости от переданного кода ошибки. Ошибки связаны с недостатком памяти, неверным размером массива, неправильными значениями и проблемами с файлами.

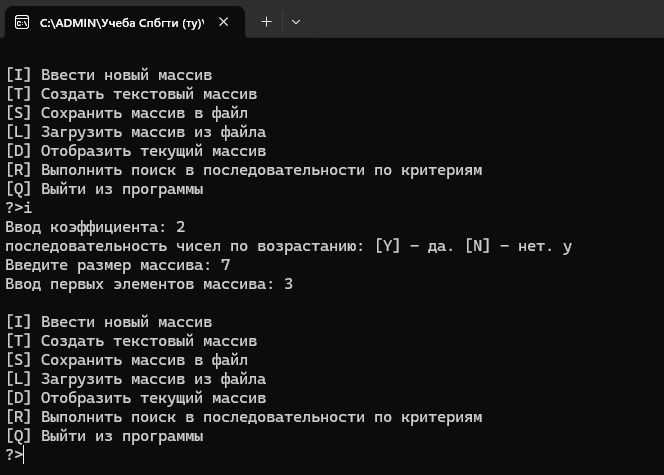
**CheckFileName**: Эта функция проверяет корректность имени файла. Она проверяет, чтобы имя не содержало недопустимых символов, таких как :, \, <, >, \*, ?, |, " и чтобы файл не начинался с символа :.

**LoadArray**: Функция загружает массив из файла с заданным именем. Сначала проверяется имя файла. Затем открывается файл и загружается размер массива. Если размер массива некорректный или возникла ошибка при чтении элементов, функция возвращает соответствующий код ошибки. Массив перезаписывается новым загруженным значением, если все прошло успешно.

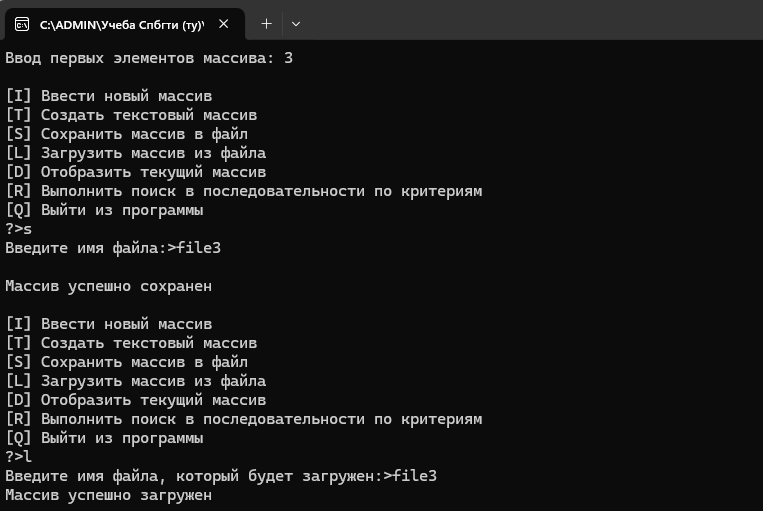
**SaveArray**: Эта функция сохраняет массив в файл с указанным именем. Перед записью она проверяет корректность имени файла и доступность для записи. Если файл существует, выполняется проверка на доступность записи (с помощью IsRewrite). Массив записывается в файл, если все условия выполнены, иначе возвращается ошибка.

**Тестирование программы:**

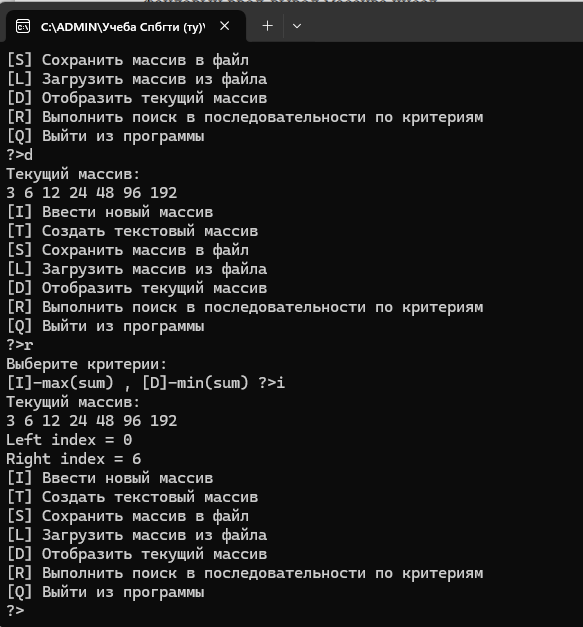
Создаем массив:



Сохраняем и выгружаем массив:



Теперь можно отобразить массив и осуществить поиск по критериям:



**Листинг программы:**

Task.h

typedef enum { /\* Список кодов ошибок \*/

errOK = 0, // Нет ошибки

errNotMemory = 1, // Ошибка: нет памяти

errArraySize = 2, // Ошибка: размер массива неверный

errArrayElement = 3, // Ошибка: элемент массива неверен

errAbsentElements = 4, // Ошибка: отсутствуют элементы

errCanNotOpenFile = 5, // Ошибка: не удается открыть файл

errCanNotSaveValue = 6, // Ошибка: не удается сохранить значение

errFileName = 7, // Ошибка: неверное имя файла

errFileExists = 8, // Ошибка: файл уже существует

errFileNotFound = 9, // Ошибка: файл не найден

errFileIsReadOnly = 10, // Ошибка: файл только для чтения

errFileIsWriteOnly = 11 // Ошибка: файл только для записи

} errCode;

//функции для работы с ошибками

errCode Error(char \*operate, errCode err);

errCode InputArray(double\* &pArray, int &Size);

errCode TestArray(double\* &pArray, int &Size);

errCode SaveArray(double\* pArray, int Size, char \*filename, bool rw);

errCode LoadArray(double\* &pArray, int &Size, char \*filename);

bool IsRewrite();

bool CheckFileName(char \*name);

void ShowArray(double \*arr, int size);

double CalcCriteria(double \*arr, int left, int right);

void Search(double \*arr, int size, bool opt, int& begin, int& end); //для поиска значений в массиве

File.cpp

#include <fstream>

#include <io.h>

#include "task.h"

using namespace std;

/\* Функция проверяет корректность имени файла filename, и

Возвращает true – если имя корректно, или false – если нет.

\*/

bool CheckFileName(char \*name)

{

char ch;

int i;

for(i=0,ch=name[0] ; ch != '\0' ; i++,ch=name[i])

{

if(ch == ':' && i != 1) return false;

if(ch == '\\' && (name[i+1] == '\0' || name[i+1] == '\\'))

return false;

if(ch == '<' || ch == '>' || ch == '\*' || ch == '?' ||

ch == '|' || ch == '\"') return false;

}

return true;

}

/\* Функция загружает массив из файла с именем filename и

диагностирует ошибки. pArray – ссылка на текущий массив,

который должен быть перезаписан, Size – ссылка на его размер

\*/

errCode LoadArray(double\* &pArray, int &Size, char \*filename)

{

double\* tmpArray = NULL;

int idx, tmpSize = 0;

if(CheckFileName(filename) == false) return errFileName;

if(\_access(filename,0) != 0) return errFileNotFound;

ifstream in(filename);

if(!in) return errCanNotOpenFile;

in >> tmpSize;

if(!in) tmpSize = 0;

if(tmpSize < 1 || tmpSize > 1000) return errArraySize;

tmpArray = new double[tmpSize];

if(tmpArray == NULL) return errNotMemory;

for(idx=0 ; idx < tmpSize ; idx++)

{

in >> tmpArray[idx];

if(!in)

{

delete[] tmpArray;

return ((in.eof()) ? errAbsentElements : errArrayElement);

}

}

if(pArray != NULL) delete[] pArray;

pArray = tmpArray;

Size = tmpSize;

return errOK;

}

/\* Функция сохраняет массив pArray размеров Size в файле с

именем filename в режиме перезаписи rw = true или отмены

записи в случае его существования rw = false.

\*/

errCode SaveArray(double \*pArray, int Size, char \*filename, bool rw)

{

if(CheckFileName(filename) == false) return errFileName;

if(\_access(filename,0) == 0)

{

if(\_access(filename,6) != 0) return errFileIsReadOnly;

if(rw == false) return errFileExists;

else if((rw = IsRewrite()) == false) return errFileExists;

}

ofstream out(filename);

if(!out) return errCanNotOpenFile;

out << Size << endl;

if(!out) return errCanNotSaveValue;

for(int idx=0 ; idx < Size ; idx++)

{

out << pArray[idx] << endl;

if(!out) return errCanNotSaveValue;

}

return errOK;

}

Inqut.cpp

using namespace std;

/\* Функция выводит на консоль массив pArray размером Size

\*/

void ShowArray(double \*pArray, int Size)

{

if(pArray != NULL || Size < 1)

{

cout << "Текущий массив:" << endl;

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

cout << pArray[i] << " ";

}

else cout << "Массив не введен";

}

/\* Функция вводит массив с консоли и диагностирует ошибки

pArray – ссылка на текущий массив, Size – ссылка на его размер

\*/

errCode InputArray(double\* &pArray, int &Size)

{

char ch = '\0';

double\* newArray = NULL;

int idx, newSize = 0;

double koeff = 0;

bool increasing = false;

cout << "Ввод коэффициента: "; // коэффициент на увеличение или уменьшение

cin >> koeff;

cin.sync();

cout << "последовательность чисел по возрастанию: [Y] - да. [N] - нет. "; // "Y" последовательность чисел на увеличение

cin >> ch;

if((ch == 'Y') || (ch == 'y')) increasing = true; // коэффициент принимает значение на увеличение

else if((ch == 'N') || (ch == 'n')) increasing = false; // коэффициент принимает значение на уменьшение

do

{

cout << "Введите размер массива: "; // сколько элементов в массиве

cin >> newSize;

cin.sync();

if(cin.fail())

{

cin.clear();

newSize = 0;

}

if(newSize < 1 || newSize > 1000)

{

cout << "Недопустимый размер массива. [A] - Прервать. [Enter] - Повторить.";

cin.get(ch);

if((ch == 'A') || (ch == 'a')) return errArraySize;

}

else ch = '\0';

}

while(ch != '\0');

newArray = new double[newSize];

if(newArray == NULL) return errNotMemory;

cout << "Ввод первых элементов массива: "; // первое значений в массиве

for(idx=0 ; idx < newSize ; idx++)

{

if(idx==0) {

cin >> newArray[idx];

cin.sync();

}

else {

if(increasing == true) newArray[idx] = newArray[idx-1] \* koeff; // заполняем массив на увеличение со 2 элемента

else if(increasing == false) newArray[idx] = newArray[idx-1] / koeff; // заполняем массив на уменьшение со 2 элемента

}

while(ch != '\0');

}

if(pArray != NULL) delete[] pArray;

pArray = newArray;

Size = newSize;

return errOK;

}

/\* Список описаний ошибок

\*/

char \*errDescription[] = {

"",

"Недостаточно памяти",

"Неверный размер массива",

"Неверное значение элемента",

"Отсутствуют элементы",

"Файл не может быть открыт",

"Значение не может быть сохранено",

"Неверное имя файла",

"Файл уже существует",

"Файл не найден",

"Доступ к файлу открыт только для чтения",

"Доступ к файлу открыт только для записи"

};

/\* Функция выводит на консоль описание ошибки

\*/

errCode Error(char \*Operate, errCode err)

{

if(err) cout << endl << Operate << " Статус: "

<< errDescription[err] << endl;

return err;

}

Lab3.cpp

#include <iostream>

#include "task.h"

using namespace std;

/\* Функция делает запрос на перезапись файла.

\*/

bool IsRewrite()

{

char ch;

cout << endl

<< "Файл уже существует. Вы хотите перезаписать его [Y/N]?";

cin >> ch;

return (ch == 'Y' || ch == 'y');

}

/\* Функция реализует консольное меню и интерфейс с пользователем.

\*/

void MainMenu(double\* &wrkArray, int& wrkSize)

{

char ch = ' ';

char filename[66];

bool bCriteria;

int idxLeft, idxRight;

do

{

cout << endl << "[I] Ввести новый массив";

cout << endl << "[T] Создать текстовый массив";

cout << endl << "[S] Сохранить массив в файл";

cout << endl << "[L] Загрузить массив из файла";

cout << endl << "[D] Отобразить текущий массив";

cout << endl << "[R] Выполнить поиск в последовательности по критериям";

cout << endl << "[Q] Выйти из программы" << endl << "?>";

cin >> ch;

switch(ch)

{

case 'I':

case 'i':

InputArray(wrkArray, wrkSize);

break;

case 'T':

case 't':

TestArray(wrkArray, wrkSize);

case 'D':

case 'd':

ShowArray(wrkArray, wrkSize);

break;

case 'S':

case 's':

cout << "Введите имя файла:>";

cin >> filename;

if (Error("Saving", SaveArray(wrkArray, wrkSize, filename, true)) == errOK)

cout << endl << "Массив успешно сохранен" << endl;

break;

case 'L':

case 'l':

cout << "Введите имя файла, который будет загружен:>";

cin >> filename;

if (Error("Loading", LoadArray(wrkArray, wrkSize, filename)) == errOK)

cout << "Массив успешно загружен" << endl;

break;

case 'R':

case 'r':

if(wrkArray == NULL)

{

cout << "Массив не введен";

break;

}

cout << "Выберите критерии:" << endl

<< "[I]-max(sum) , [D]-min(sum) ?>";

cin >> ch;

if(ch == 'I' || ch == 'i') bCriteria = true;

else if(ch == 'D' || ch == 'd') bCriteria = false;

else { cout << "Неверный вариант: " << ch;

ch = ' ';

break;

}

Search(wrkArray, wrkSize, bCriteria, idxLeft, idxRight);

ShowArray(wrkArray, wrkSize);

cout << endl << "Left index = " << idxLeft

<< endl << "Right index = " << idxRight;

break;

default:

cout << "Неправильный пункт меню";

}

cin.sync();

}

while(ch != 'Q' && ch != 'q');

}

void main(void) // Главная функция

{

std::setlocale(LC\_ALL, "ru\_RU.UTF-8"); // Устанавливаем локаль для работы с кириллицей

std::wcout.imbue(std::locale("ru\_RU.UTF-8")); // Настройка для вывода русских символов

double\* myArray = NULL; // Текущий рабочий массив

int arrSize = 0; // Размер текущего массива

MainMenu(myArray, arrSize);

}

Solve.cpp

#include <float.h>

#include <stddef.h>

#include "task.h"

/\* Функция загружает тестовый массив в качестве текущего массива

\*/

errCode TestArray(double\* &pArray, int &Size)

{

static double test[] =

{ 2.1,4.2,8.4,16.8,33.6,67.2,134.4,268.8,537.6,1075.2,2150.4,4300.8 };

if(pArray != NULL) delete[] pArray;

Size = sizeof(test) / sizeof(double);

pArray = new double[Size];

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

pArray[i] = test[i];

return errOK;

}

/\* Функция вычисляет сумму последовательности элементов

в заданном массиве arr от индекса left до индекса right

\*/

double CalcCriteria(double \*arr, int left, int right)

{

double sum = 0;

for(int j=left ; j <= right ; j++)

sum += arr[j];

return sum;

}

/\* Функция ищет в массиве arr размерностью size такую непрерывную

последовательность элементов, начинающуюся с индекса begin и

заканчивающуюся индексом end, которая имеет максимальную при

opt = true или минимальную при opt = false сумму элементов.

\*/

void Search(double \*arr, int size, bool opt, int& begin, int& end)

{

int i,m;

double sum;

double criteria = ((opt) ? -DBL\_MAX : DBL\_MAX);

for(i=0 ; i <= size-2 ; i++)

{

for(m=2 ; m <= size-i ; m++)

{

sum = CalcCriteria(arr, i, (i + m - 1));

if((opt) ? sum > criteria : sum < criteria)

{

criteria = sum;

begin = i;

end = i + m - 1;

}

}

}

}