Кафедра «Высшая математика»

Е.В. Кадура

Операционные   
системы

Методические указания   
по выполнению курсовой работы

Хабаровск

2020

Оглавление

[ВВЕДЕНИЕ 3](#_Toc57200489)

[1 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ 4](#_Toc57200490)

[2 РЕАЛИЗАЦИЯ ПРОГРАММЫ-ДИСПЕТЧЕРА НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C++. 5](#_Toc57200491)

[2.1 Назначение разработки 5](#_Toc57200492)

[2.2 Описание интерфейса создаваемого приложения 5](#_Toc57200493)

[2.3 Описание системы меню 5](#_Toc57200494)

[2.4 Описание вспомогательных форм 5](#_Toc57200495)

[3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 6](#_Toc57200496)

[3.1 Общие сведения о программном продукте 6](#_Toc57200497)

[3.2 Описание установки 6](#_Toc57200498)

[3.3 Описание запуска 6](#_Toc57200499)

[3.4 Инструкции по работе 6](#_Toc57200500)

[3.5 Сообщения пользователю 7](#_Toc57200501)

[ЗАКЛЮЧЕНИЕ 8](#_Toc57200502)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 9](#_Toc57200503)

[ПРИЛОЖЕНИЕ 1. ИСХОДНЫЙ КОД ПРОГРАММЫ 11](#_Toc57200504)

# ВВЕДЕНИЕ

Целью курсовой работы является закрепление изученного материала в ходе разработки программы – диспетчера операционной системы Windows, позволяющей просматривать текстовые файлы, расположенные в дереве каталогов файловой системы локальных дисков.

Разработка приложения будет осуществляться в среде Microsoft Visual Studio на языке программирования C++, с использованием платформы программирования Qt для построения графического пользовательского интерфейса. Выбор языка и платформы обусловлен необходимостью освоить совмещение популярной среды среды разработки Microsoft Visual Studio и мощной сторонней библиотеки классов Qt, значительно превосходящей по возможностям и удобству использования библиотеку MFC.

# ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

делается обзор литературных источников, который должен содержать краткую оценку изложенных материалов и принципов программирования.

Теоретический раздел курсовой работы пишут в соответствии с вариантом

# РЕАЛИЗАЦИЯ ПРОГРАММЫ-ДИСПЕТЧЕРА НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C++.

## Назначение разработки

В разделе «Назначение разработки» указать, какие задачи будут реализованы и цели реализации. Сделать описание системы меню приложения. Указать типы объектов, которые могут использоваться при загрузке файла созданного приложения.

## Описание интерфейса создаваемого приложения

В разделе «Описание интерфейса создаваемого приложения» необходимо провести теоретический анализ реализуемой задачи и выбрать для практической реализации приемлемый вариант. Указать какие средства программирования облегчат решение задачи (например, использование библиотеки .NET, использование классов и др.). Указать способы взаимодействия с пользователем, которые использовались при создании приложения. Привести скриншоты основных окон создаваемого приложения и описание к ним.

## Описание системы меню

В разделе «Описание системы меню» привести скелетную структуру меню приложения. Пояснить логику создания такого меню

## Описание вспомогательных форм

В разделе «Описание вспомогательных форм» привести скриншоты вспомогательных форм и дать подробное описание этих форм до и после осуществления какого-либо действия.

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

**Рекомендации по написанию подобной программной документации:**

• учитывайте интересы пользователей – руководство должно содержать все инструкции, необходимые пользователю;

• излагайте ясно, используйте короткие предложения;

• избегайте технического жаргона и узкоспециальной терминологии, если все же необходимо использовать некоторые термины, то их следует пояснить;

• будьте точны и рациональны – длинные и запутанные руководства обычно никто не читает, например, лучше привести рисунок формы, чем долго ее описывать.

## Общие сведения о программном продукте

1. Раздел «Общие сведения о программе» обычно содержит наименование программного продукта, краткое описание его функций, реализованных методов и возможных областей применения.

## Описание установки

2. Раздел «Описание установки» обычно содержит подробное описание действий по установке программного продукта и сообщений, которые при этом могут быть получены.

## Описание запуска

3. В разделе «Описание запуска», как правило, описаны действия по запуску программного продукта и сообщений, которые при этом могут быть получены.

## Инструкции по работе

4. Раздел «Инструкции по работе» (или описание пользовательского интерфейса) обычно содержит описание режимов работы, форматов ввода-вывода информации и возможных настроек.

## Сообщения пользователю

5. Раздел «Сообщения пользователю» должен содержать перечень возможных сообщений, описание их содержания и действий, которые необходимо предпринять по этим сообщениям.

# ЗАКЛЮЧЕНИЕ

*В результате работы была спроектирована и разработана программа, позволяющая работать с квадратными матрицами, вводимыми пользователем и вычислять произведение положительных элементов их главных диагоналей.*

*В ходе работы над проектом получены практические навыки разработки программ на языке высокого уровня C. Изучено множество аспектов разработки программ на этом языке – в первую очередь, организация динамических структур данных и управление памятью.*

*На базе практического применения, закреплены базовые знания, полученные ранее, при изучении теории.*

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Стивен Прата Язык программирования C. Лекции и упражнения / Стивен Прата. - М.: Вильямс, 2015. - 928 c.

2. Подбельский В.В, Фомин С.С. Программирование на языке C / М.: «Финансы и статистика». 2003.

3. Дейтел Х.М., Дейтел П.Дж. Как программировать на C / М: «БИНОМ», 2000.

4. Шилдт, Г. Справочник по C. Диасофт М.: 2000.

5. ГОСТ 2.105-95 Общие требования к текстовым документам.

6.

7.

8.

9.

10.

11.

12.

13.

14.

15.

**ПРИЛОЖЕНИЯ**

# ПРИЛОЖЕНИЕ 1. ИСХОДНЫЙ КОД ПРОГРАММЫ

#include "pch.h"

#include <stdlib.h>

#include <stdio.h>

#define A\_SIZE 5

#define B\_SIZE 7

#define C\_SIZE 4

// Предварительное объявление функции

int main\_positive\_product(int\* matrix, int dimSize);

// Основная функция - точка входа программы

int main()

{

// Выделение памяти под массивы для

// хранения элементов исходных матриц

int\* A = (int\*)malloc(sizeof(int) \* A\_SIZE \* A\_SIZE);

int\* B = (int\*)malloc(sizeof(int) \* B\_SIZE \* B\_SIZE);

int\* C = (int\*)malloc(sizeof(int) \* C\_SIZE \* C\_SIZE);

// Ввод матрицы A

printf("A[%d][%d]:\n", A\_SIZE, A\_SIZE);

for (int i = 0; i < A\_SIZE; ++i)

{

for (int j = 0; j < A\_SIZE; ++j)

{

printf("input A[%d][%d]:", i, j);

// Смещаем При вводе указатель на &A[i][j]

scanf("%d", A + A\_SIZE \* i + j);

}

}

// Ввод матрицы B

printf("B[%d][%d]:\n", B\_SIZE, B\_SIZE);

for (int i = 0; i < B\_SIZE; ++i)

{

for (int j = 0; j < B\_SIZE; ++j)

{

printf("input B[%d][%d]:", i, j);

// Смещаем При вводе указатель на &B[i][j]

scanf("%d", B + B\_SIZE \* i + j);

}

}

// Ввод матрицы C

printf("C[%d][%d]:\n", C\_SIZE, C\_SIZE);

for (int i = 0; i < C\_SIZE; ++i)

{

for (int j = 0; j < C\_SIZE; ++j)

{

printf("input C[%d][%d]:", i, j);

// Смещаем При вводе указатель на &C[i][j]

scanf("%d", C + C\_SIZE \* i + j);

}

}

// Вычисляем произведения положительных элементов

// главной диагонали каждой из исходных матриц

int P\_A = main\_positive\_product(A, A\_SIZE);

int P\_B = main\_positive\_product(B, B\_SIZE);

int P\_C = main\_positive\_product(C, C\_SIZE);

// Вычисляем результат выражения

int P = P\_A + P\_B - P\_C;

// Выводим результат

printf("%d + %d - %d = %d\n", P\_A, P\_B, P\_C, P);

// Ожидаем реакции пользователя

printf("Press any key to finish\n");

getchar();

// Освобождаем выделенную память

free(A);

free(B);

free(C);

}

/\* Возвращает произведение положительных

элементов главной диагонали квадратной матрицы

Параметры:

matrix - указатель на первый элемент массива,

содержащего элементы квадратной матрицы

dimSize - количество строк/столбцов матрицы

\*/

int main\_positive\_product(int\* matrix, int dimSize)

{

int result = 1;

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

{

int m = matrix[dimSize \* i + i];

if (m > 0)

{

result \*= m;

}

}

return result;

}