МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«Гомельский государственный университет имени Франциска Скорины»

Факультет физики и информационных технологий

Кафедра общей физики

ОТЧЕТ

по учебной ознакомительной практике студента 1 курса

специальности 1-1-39 03 02 Программируемые мобильные системы

на (в) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Срок практики: с \_\_\_\_\_\_\_\_\_\_\_ по \_\_\_\_\_\_\_\_\_\_\_

Студент группы МС-12 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Жихарева А.Д.

(подпись)

Руководитель практики  
от университета \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Грищенко В.В.

(подпись)

Гомель 2022

**РЕФЕРАТ**

**Учебной ознакомительной практики:** 23 страницы, 23 рисунка.

**Ключевые слова:** Си, задание, разработка программы.

**Цель учебной ознакомительной практики:** получение опыта программирования на C путем решения задач учебной ознакомительной практики.

**Метод исследования:** теоретический метод, эксперимент.

**Содержание**

|  |  |
| --- | --- |
| Реферат………………………………………………………………………….  1. Задание 1 ……………………………………………………………………   * 1. Условие ………………………………………………………….   2. Блок-схема ………………………………………………………..   3. Описание программы …………………………………………….   2. Задание 2 ……………………………………………………………………   * 1. Условие 1 …………………………………………………………   2. Блок-схема ………………………………………………………..   3. Описание программы …………………………………………….   4. Условие 2 …………………………………………………………   5. Описание программы …………………………………………….   6. Условие 3 …………………………………………………………   7. Описание программы …………………………………………….   3. Задание 3 ……………………………………………………………………   * 1. Условие ………………………………………………………….   2. Блок-схема ………………………………………………………..   3. Описание программы …………………………………………….   4. Задание 4 ……………………………………………………………………   * 1. Условие ………………………………………………………….   2. Блок-схема ………………………………………………………..   3. Описание программы …………………………………………….   Заключение …………………………………………………………………..  Список источников …………………………………………………………….  Приложение 1 ………………………………………………………………….  Приложение 2 …………………………………………………………………..  Приложение 3 ………………………………………………………………….  Приложение 4 …………………………………………………………………..  Приложение 4.1 ………………………………………………………………..  Приложение 4.2 ……………………………………………………………….  Приложение 5 ………………………………………………………………….  Приложение 6 ………………………………………………………………….. | 2  4  4  4  5  5  5  6  6  7  7  7  8  9  9  9  10  10  10  11  11  14  15  16  17  18  19  20  20  21  22 |

**1.Задание 1**

**1.1 Условие**

Разработать, используя цикл while и не вводя массивы, программу обработки последовательности чисел. Подсчитать количество отрицательных нечетных чисел последовательности. Окончание ввода – число, кратное 4.

**1.2 Блок-Схема**

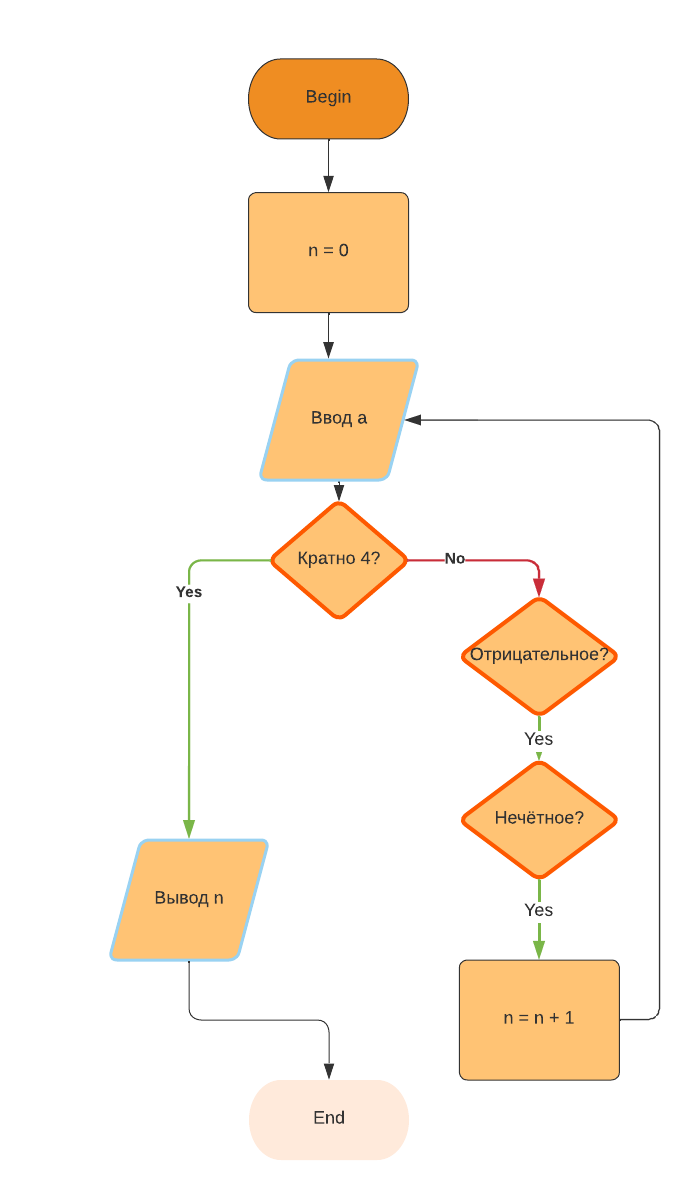


Рисунок 1 – Блок-схема

**1.3 Описание программы**

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Цикл обработки чисел

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – Вывод программы

**2. Задача 2**

Разработать программу обработки одномерного массива.

**2.1 Условие 1**

Дан массив A[n]. Найти произведение элементов, расположенных между первым по порядку минимальным и последним по порядку максимальным элементами массива.

* 1. **Блок-схема**

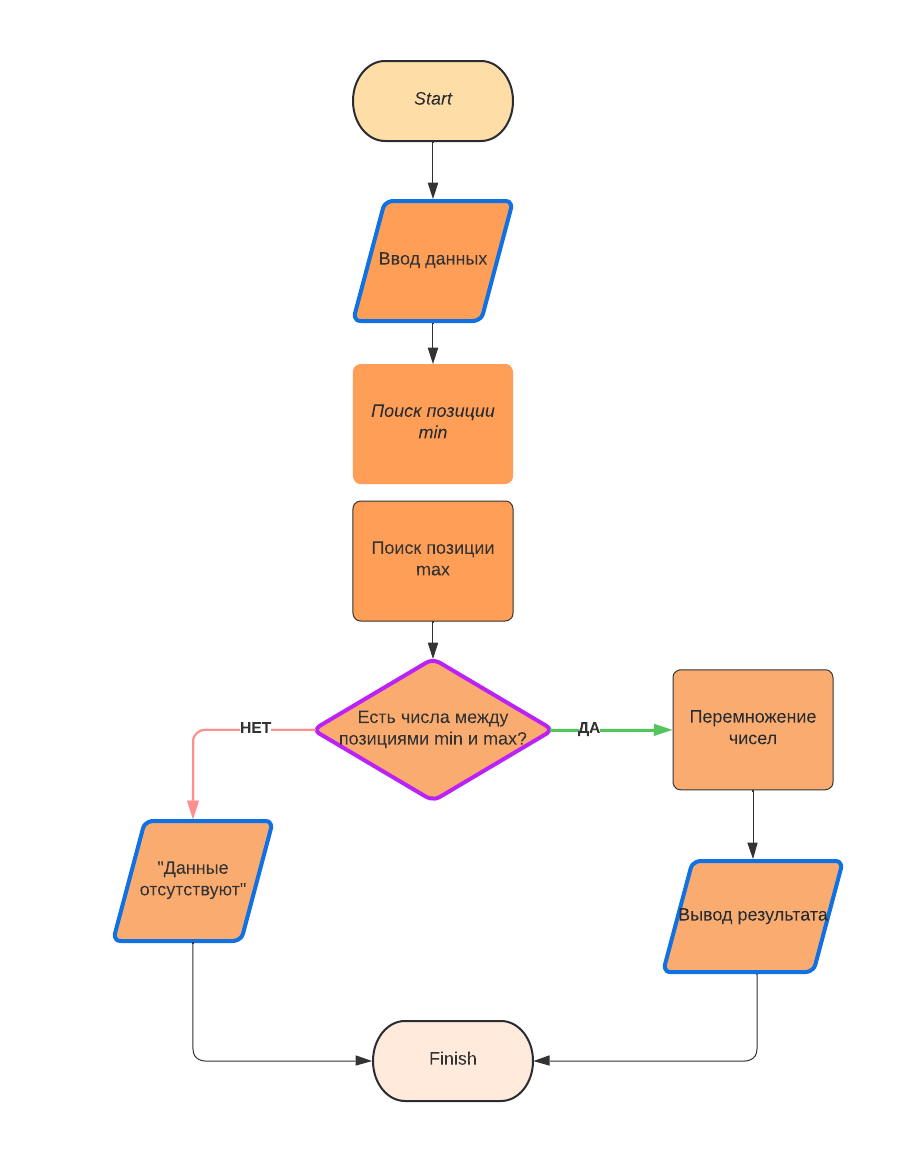


Рисунок 4 – Блок-схема

**2.3** **Описание программы**

|  |  |
| --- | --- |
|  |  |

Рисунок 5 – Циклы поиска минимума и максимума

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 – Работа с полученными данными

**2.4 Условие 2**

Выполнить условие 1, используя для доступа к элементам массива указатели (операцию [] использовать нельзя).

**2.5 Описание программы**

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание

Рисунок 7 – Циклы поиска минимума и максимума

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8 – Работа с полученными данными

**2.6 Условие 3**

Выполнить условие 1 в виде двух единиц компиляции: один c–файл с функцией main(), второй c–файл с разработанной функцией для обработки массива. Исходные данные должны вводиться из файла с именем, которое задается в командной строке, и расширением in, в функции main(). Результат работы программы должен выводиться в файл с именем входного файла с расширением out в функции main(). Память под массив должна выделяться динамически.

**2.8 Описание программы**

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9 – Создание и подключение библиотеки с вычислениями

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10 – Ввод названия файла

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 11 – Запись конечных данных в файл

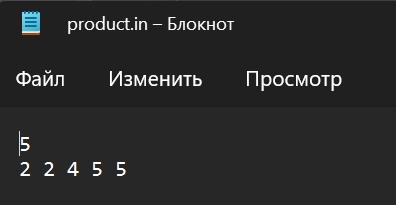


Рисунок 12 – Файл с исходными данными

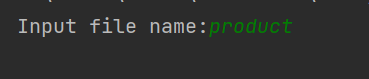


Рисунок 13 – Ввод имени файла

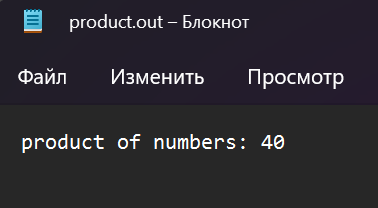


Рисунок 14 – Файл с конечными данными

**3. Задание 3**

**3.1 Условие**

Дано слово. Заменить в слове все маленькие латинские буквы от “a” до “y” на следующие по алфавиту.

**3.2 Блок-схема**

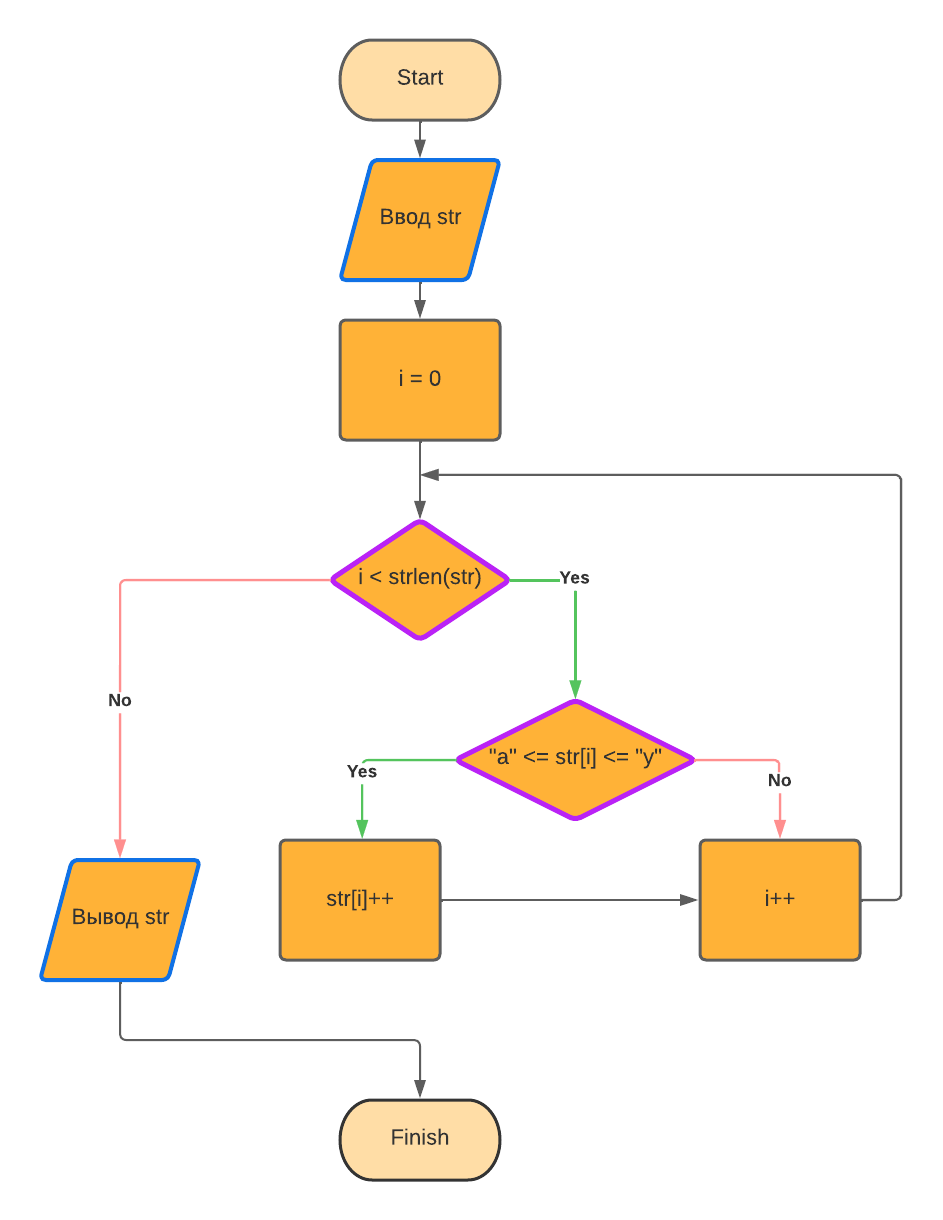


Рисунок 15 – Блок-схема

**3.3 Описание программы**

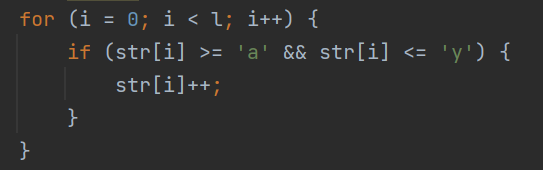


Рисунок 16 – Цикл обработки символьной информации

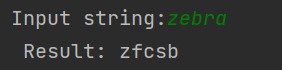


Рисунок 17 – Вывод программы

**4. Задание 4**

**4.1 Условие**

Для заданного объекта описать шаблон структуры и выполнить следующие действия: описать одномерный массив структурных переменных, заполнив его с клавиатуры, выполнить требуемые действия в соответствии с заданием 1, вывести на экран полученные результаты; ввести размерность массива, динамически создать одномерный массив структурных переменных, заполнить его с клавиатуры, выполнить требуемые действия в соответствии с заданием 2, вывести на экран полученные результаты, удалить массив. Для доступа к элементам динамического массива использовать указатели (операцию [] использовать нельзя).

Объект – дом (название улицы; номер дома; тип: обычный, кооперативный, малосемейный, другое). Множество данных – дома домоуправления. Задание: 1 – отсортировать дома по убыванию названия улицы, дома одной улицы должны быть отсортированы по возрастанию номера; 2 – найти количество различных типов домов, номер которых находится в заданном диапазоне.

* 1. **Блок-схема**

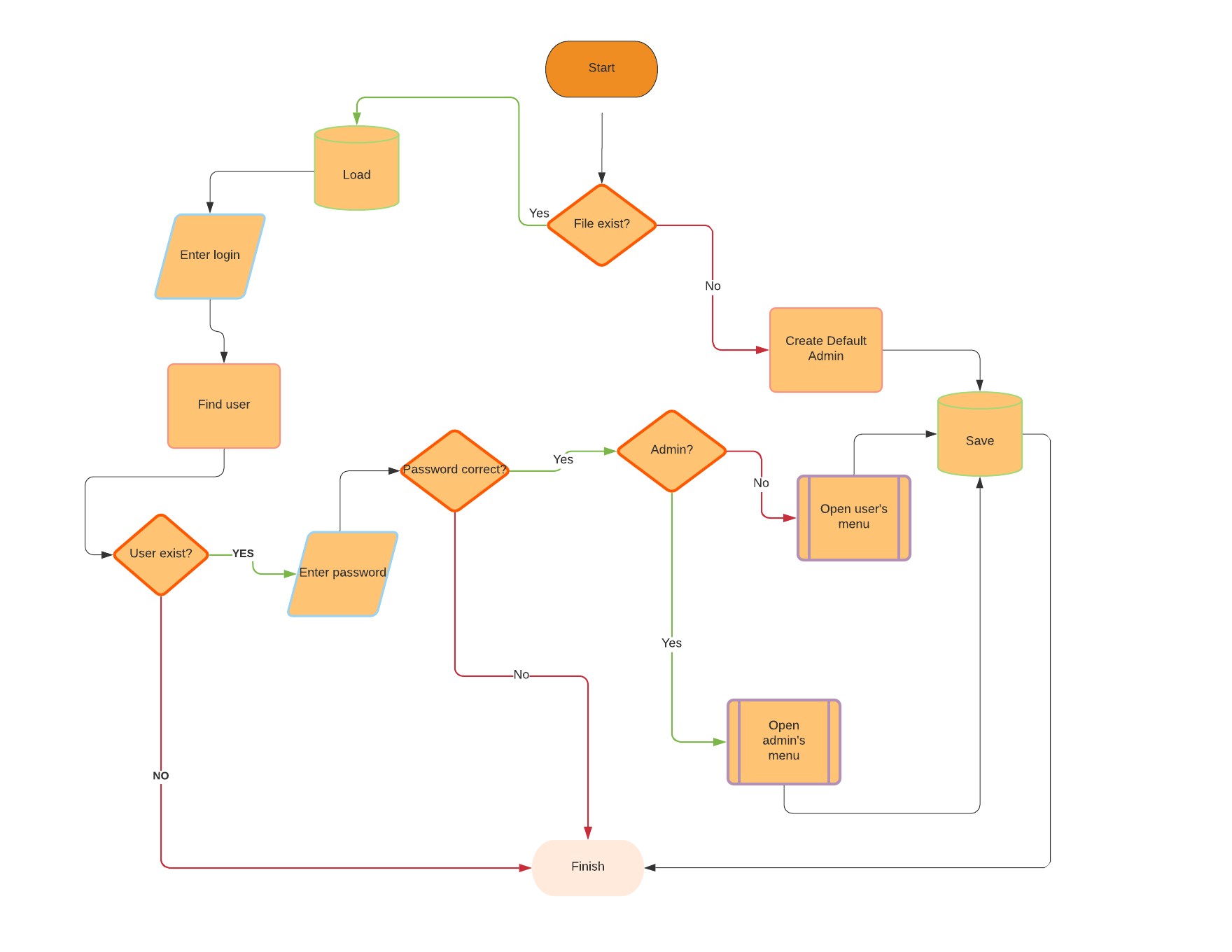


Рисунок 18 – Блок-схема

* 1. **Описание программы**

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 19 – Функция сортировки домов на одинаковых улицах по возрастанию номера

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 20 – Функция нахождения количества разных типов домов в заданном промежутке их номеров

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 21 – Ввод данных

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 22 – Задание 1

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 23 – Задание 2

**Заключение**

Все задания учебной ознакомительной практики были выполнены и представлены в отчёте. На каждое задание была приведена блок-схема, которая упрощает понимание логики программы. Также все значимые функции программы были представлены в рисунках.

В ходе работы были закреплены знания о базовых функциях языка Си, а также был приобретён практический опыт в программировании на данном языке и были изучены на практике функции среднего уровня сложности.

**Список использованных источников**

1. https://ru.stackoverflow.com/questions/
2. https://www.cyberforum.ru/c-beginners/
3. <https://studassistent.ru/c>
4. <http://www.c-cpp.ru/content/calloc>
5. <http://all-ht.ru/inf/prog/c/func/>
6. <https://prog-cpp.ru/c-files/>
7. http://www.codenet.ru/progr/cpp/1/h17.php

**Приложение 1**

#include <stdio.h>

int main() {

printf("Input a: ");

int n = 0;

int a = 1;

while (a % 4 != 0) {

scanf("%d", &a);

if (a < 0 && a % 2 != 0) {

n = n + 1;

}

}

printf("Number of negative odd numbers: %d", n);

}

**Приложение 2**

#include <stdio.h>

#include <stdlib.h>

#include <windef.h>

int main() {

int a[100];

int n, i;

printf("Input n: ");

scanf("%d", &n);

printf("Input massive: ");

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

scanf("%d", &a[i]);

}

int imin = 0;

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

if (a[i] < a[imin]) {

imin = i;

}

}

int imax = 0;

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

if (a[i] >= a[imax]) {

imax = i;

}

}

int c = 1;

if (abs(imax - imin) <= 1) {

printf("Numbers not founded");

} else {

for (i = min(imin, imax) + 1; i < max(imin,imax); i++) {

c = c \* a[i];

}

printf("The product of numbers: %d", c);

}

}

**Приложение 3**

#include <stdio.h>

#include <stdlib.h>

#include <windef.h>

int main() {

int \*a;

int n, i;

int \*ptr;

printf("Input n: ");

scanf("%d", &n);

a = (int \*) calloc(n, sizeof(int));

printf("Input massive: ");

for (ptr = a; ptr < a + n; ptr++) {

scanf("%d", ptr);

}

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

printf("%d ", a[i]);

}

int \*imin = a;

for (ptr = a; ptr < a + n; ptr++) {

if (\*ptr < \*imin) {

imin = ptr;

}

}

int \*imax = a;

for (ptr = a; ptr < a + n; ptr++) {

if (\*ptr >= \*imax) {

imax = ptr;

}

}

int c = 1;

if (abs(imax - imin) <= 1) {

printf("Numbers not founded");

} else {

for (ptr = min(imin, imax) + 1; ptr < max(imin, imax); ptr++) {

c = c \* \*ptr;

}

printf("The product of numbers: %d", c);

}

}

**Приложение 4**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "product.h"

int main() {

int \*a;

int n;

int \*ptr;

char name[20];

char inname[20];

char outname[20];

printf("Input file name:");

scanf("%s", name);

strcpy(inname, name);

strcat(inname, ".in");

FILE \*fl;

fl = fopen(inname, "r");

fscanf(fl, "%d", &n);

a = (int \*) calloc(n, sizeof(int));

for (ptr = a; ptr < a + n; ptr++) {

fscanf(fl, "%d", ptr);

}

fclose(fl);

int c = product(a, n);

strcpy(outname, name);

strcat(outname, ".out");

fl = fopen(outname, "w");

if (c == 0) {

fprintf(fl, "no elements");

} else {

fprintf(fl, "product of numbers: %d", c);

}

fclose(fl);

}

**Приложение 4.1**

#ifndef LAB\_PRODUCT\_H

#define LAB\_PRODUCT\_H

#endif //LAB\_PRODUCT\_H

extern int product(int \*a, int n);

**Приложение 4.2**

#include <stdlib.h>

#include <windef.h>

#include "product.h"

int product(int \*a, int n){

int \*ptr;

int \*imin = a;

for (ptr = a; ptr < a + n; ptr++) {

if (\*ptr < \*imin) {

imin = ptr;

}

}

int \*imax = a;

for (ptr = a; ptr < a + n; ptr++) {

if (\*ptr >= \*imax) {

imax = ptr;

}

}

int c = 1;

if (abs(imax - imin) <= 1) {

return 0;

}

else {

for (ptr = min(imin, imax) + 1; ptr < max(imin, imax); ptr++) {

c = c \* \*ptr;

}

return c;

}

}

**Приложение 5**

#include <stdio.h>

#include <string.h>

int main() {

char str[100];

int i, l;

printf("Input string: ");

scanf("%s", str);

l = strlen(str);

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

if (str[i] >= 'a' && str[i] <= 'y') {

str[i]++;

}

}

printf("Result: %s", str);

}

**Приложение 6**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

const int house\_type = 4;

static char \*HouseTypeNames[] = {

"other",

"regular",

"cooperative",

"small-family"

};

struct house {

char street[10000000];

int number;

int type;

};

int compareHouses(const void \*a, const void \*b) {

int s = strcmp(((struct house \*) a)->street, ((struct house \*) b)->street);

if (s == 0) {

return ((struct house \*) a)->number - ((struct house \*) b)->number;

}

return s;

}

struct house \* EnterHouses(int houses\_num) {

int i;

struct house \*houses = (struct house \*) calloc(houses\_num, sizeof(struct house));

for (i = 0; i < houses\_num; i++) {

struct house \*hs = houses + i;

printf("Enter house street: ");

scanf("%s", hs->street);

printf("Enter house number: ");

scanf("%d", &hs->number);

printf("Choose house type(1 - regular, 2 - cooperative, 3 - small-family, 0 - other):");

scanf("%d", &hs->type);

}

return houses;

}

void PrintHouses(struct house \* houses, int houses\_num) {

printf("\n");

printf("+---+--------------+--------------+----------------+\n");

printf("| N | Street | House number | House type |\n");

printf("+---+--------------+--------------+----------------+\n");

for (int i = 0; i < houses\_num; i++) {

struct house \*hs = houses + i;

printf("|%3d|%14s|%14d|%16s|\n",

i + 1, hs->street, hs->number, HouseTypeNames[hs->type]);

}

printf("+---+--------------+--------------+----------------+\n");

}

void FindHouseTypes(struct house \* houses, int houses\_num) {

int a, b;

printf("Enter range of house numbers: from ");

scanf("%d", &a);

printf(" to ");

scanf("%d", &b);

int \*n = (int \*) calloc(house\_type, sizeof(int));

for (int i = 0; i < house\_type; i++) {

\*(n + i) = 0;

}

int k = 0;

for (int i = 0; i < houses\_num; i++) {

if ((houses + i) -> number <= b && (houses + i) -> number >= a) {

\*(n + (houses + i) -> type) = 1;

}

}

for (int i = 0; i < house\_type; i++) {

k += \*(n + i);

}

printf("Number of different house types: %d", k);

free(n);

}

int main() {

struct house \* houses;

int houses\_num;

printf("Enter number of houses: ");

scanf("%d", &houses\_num);

houses = EnterHouses(houses\_num);

qsort(houses, houses\_num, sizeof(struct house), compareHouses);

PrintHouses(houses, houses\_num);

FindHouseTypes(houses, houses\_num);

free(houses);

}