**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 8

по дисциплине: «Основы алгоритмизации и программирования»

**Студент группы № 219/2**

**Клопова Ульяна**

**Вариант 15**

**Преподаватель: Рябова В.В.**

Санкт-Петербург

2024 г.

**Задание 1.** Создать функцию, которая возвращает в рублях сумму, заданную в долларах. Для создаваемой функции: подобрать имя; указать тип функции; выбрать имена и типы входных параметров; описать тело функции с обязательным оператором в конце; в главной программе вызвать созданную функцию два раза с различными входными данными. Вывести результаты в главной программе.

#include <iostream>

using namespace std;

void money(unsigned lol) {

int c;

const int a = 91;

c = lol \* a;

cout << "Рубли: " << c << endl;

}

int main()

{

int b, c=0;

setlocale(0, "");

do {

cout << "Введите доллары:" << endl;

cin >> b;

money(b);

c++;

} while (c != 2);

}

**Запуск 1.**

**Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.**

Результат при b = 4 и 3

**Запуск 2.**

**Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.**

Результат при b = 53 и 7

**Запуск 3.**

**Изображение выглядит как текст, Шрифт, снимок экрана, число

Контент, сгенерированный ИИ, может содержать ошибки.**

Результат при b = 6 и 23

**Задание 2.** Реализовать функцию. Функция вычисляет расстояние до начала координат от точки на плоскости по заданным её координатам x и y. В главной программе задано две точки. Найти их расстояния до начала координат, вызвав функцию 2 раза.

#include <iostream>

#include <cmath>

using namespace std;

double meh (double lelx,double lely) {

double rez;

return rez = sqrt((lelx \* lelx) + (lely \* lely));

}

int main()

{

double b, c;

int clik=0;

setlocale(0, "");

do{

cout << "Введите 2 точки (x, y)" << endl;

cin >> b;

cin >> c;

cout << "Результат: " << meh(b, c) << endl;

clik++;

} while (clik != 2);

}

**Запуск 1.**

**Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.**

Результат при b = 3 и c = 4.3

b = 2 и c = 6

**Запуск 2.**

**Изображение выглядит как текст, Шрифт, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.**

Результат при b = 0.5 и c = 4

b = 3 и c = 9

**Запуск 3.**

**Изображение выглядит как текст, Шрифт, снимок экрана, число

Контент, сгенерированный ИИ, может содержать ошибки.**

Результат при b = 1 и с =34

b = 6.7 и с = 0.4

**Задание 3.** Описать функцию RingS(R1,R2) вещественного типа, находящую площадь кольца, заключенного между двумя окружностями с общим центром и радиусами R1 иR2 (R1 и R2 — вещественные, R1 > R2). С ее помощью найти площади трех колец, для которых даны внешние и внутренние радиусы. Воспользоваться формулой площади круга радиуса R: S = р\*R2. В качестве значения р использовать 3.14.

#include <iostream>

using namespace std;

double meh(double lelx, double lely) {

const double pi = 3.14;

double ar = pi \* lelx \* lelx;

double ar2 = pi \* lely \* lely;

return ar - ar2;

}

int main()

{

double r1[3], r2[3];

setlocale(0, "");

cout << "Введите внешний радиус R1 для кольца где первый ввод = первому вводу внутреннего радиуса = 1 кольцо: "<<endl;

for (int i = 0; i < 3; i++) {

cin >> r1[i];

}

cout << "Введите внутренний радиус R2 для кольца : "<<endl;

for (int i = 0; i < 3; i++) {

cin >> r2[i];

}

for (int i = 0; i < 3; i++) {

if (r1[i] > r2[i]) {

double area = meh(r1[i], r2[i]);

cout << "Площадь кольца " << i + 1 << ": " << area << endl;

}

else {

cout << "Ошибка: внешний радиус R1 должен быть больше внутреннего радиуса R2 для кольца " << i + 1 << endl;

}

}

}

**Запуск 1.**

**Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.**

Результат при r1 = 2, 3, 4, и r2 = 1,2,3

**Запуск 2.**

**Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.**

Результат при r1 = 1, 2, 3, и r2 = 4,5,6

**Запуск 3.**

**Изображение выглядит как текст, Шрифт, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.**

Результат при r1 = 2, 1, 3, и r2 = 1,3,2

**Задание 4.** Оформить функцию поиска среднего арифметического положительных элементов массива. В главной программе дано 3 одномерных массива a,b,c длиной 10 элементов каждый. Применить функцию для каждого из 3-х заданных массивов. (в функции не должно быть операторов ввода или вывода)

#include <iostream>

#include <iomanip>

#include <cstdlib>

#include <ctime>

using namespace std;

void aboba(int a[], int siz) {

for (int i{}; i < siz; i++) {

a[i] = rand() % 210 - 18;

}

}

void bubus(int a[], int siz) {

for (int i{}; i < siz; i++) {

cout << setw(5) << a[i];

}

}

double aaa(const int ab[], int shish) {

double um = 0;

int count = 0;

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

if (ab[i] > 0) {

um += ab[i];

count++;

}

}

return (count > 0) ? um / count : 0.0;

}

int main()

{

setlocale(0, "");

srand(time(0));

const int sos = 10;

int lal[sos]{}, lala[sos]{}, alal[sos]{}, sum = 0, x;

cout << "Массивы: \n";

for (int i = 0; i < 3; i++) {

if (i == 0) {

aboba(lal, sos);

cout << endl << endl;

cout << "Массив 1:" << endl;

bubus(lal, sos);

cout << " Среднее арифметическое положительных элементов: " << aaa(lal, sos) << endl;

}

else if (i == 1) {

aboba(lala, sos);

cout << endl << endl;

cout << "Массив 2:" << endl;

bubus(lala, sos);

cout << " Среднее арифметическое положительных элементов: " << aaa(lala, sos) << endl;

}

else if (i == 2) {

aboba(alal, sos);

cout << endl << endl;

cout << "Массив 3:" << endl;

bubus(alal, sos);

cout << " Среднее арифметическое положительных элементов: " << aaa(alal, sos) << endl;

}

}

}

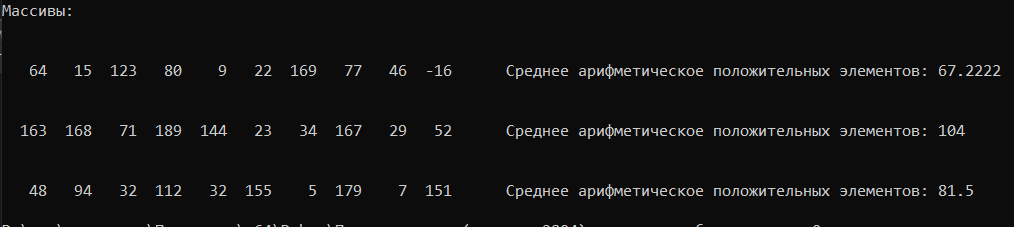
**Запуск 1.**

**Изображение выглядит как снимок экрана, текст, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.**

Результат при первом запуске

**Запуск 2.**

****

Результат при втором запуске

**Запуск 3.**

**Изображение выглядит как снимок экрана, текст, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.**

Результат при третьем запуске

**Задание 5.** Оформить функцию поиска произведение положительных элементов массива, стоящих на четных местах (использовать шаг цикла ≠1), В главной программе Дано 3 одномерных массива a,b,c длиной 30 элементов каждый. Применить функцию для каждого из 3-х заданных массивов. найти произведение найденных произведений элементов. (в функции не должно быть операторов ввода или вывода

#include <iostream>

#include <iomanip>

#include <cstdlib>

#include <ctime>

using namespace std;

void aboba(int a[], int siz) {

for (int i{}; i < siz; i++) {

a[i] = rand() % 21 - 15;

}

}

void bubus(int a[], int siz) {

for (int i{}; i < siz; i++) {

cout << setw(5) << a[i];

}

}

int aaa(const int ab[], int shish) {

int um = 1;

bool f = false;

for (int i = 1; i < shish; i+=2) {

if (ab[i] > 0) {

um \*= ab[i];

f = true;

}

}

return f ? um :0;

}

int main()

{

setlocale(0, "");

srand(time(0));

const int sos = 30;

int lal[sos]{}, al=0, lala[sos]{},ja = 0, alal[sos]{}, aa =0, bip =0;

cout << "Массивы: \n";

for (int i = 0; i < 3; i++) {

if (i == 0) {

aboba(lal, sos);

cout << endl << endl;

cout << "Массив 1:" << endl;

bubus(lal, sos);

al = aaa(lal, sos);

cout << " произведение положительных элементов массива: " << al << endl;

}

else if (i == 1) {

aboba(lala, sos);

cout << endl << endl;

cout << "Массив 2:" << endl;

bubus(lala, sos);

ja = aaa(lala, sos);

cout << " произведение положительных элементов массива: " << ja << endl;

}

else if (i == 2) {

aboba(alal, sos);

cout << endl << endl;

cout << "Массив 3:" << endl;

bubus(alal, sos);

aa = aaa(alal, sos);

cout << " произведение положительных элементов массива: " << aa << endl;

}

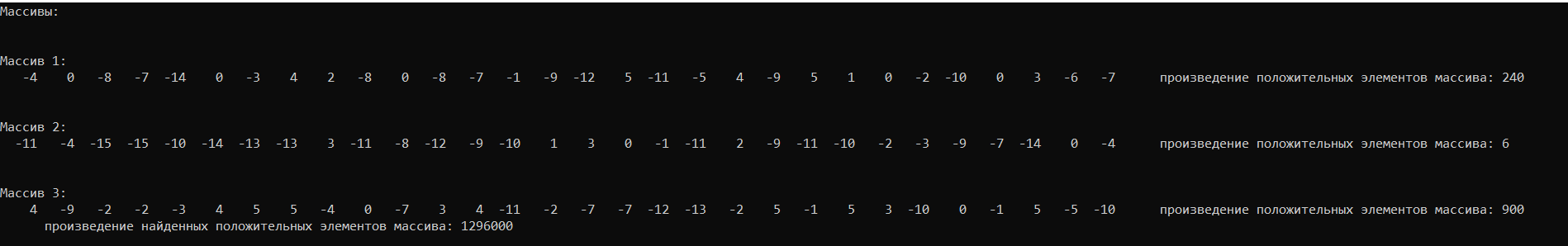
}

bip = al \* ja \* aa;

cout << " произведение найденных положительных элементов массива: " << bip << endl;

}

**Запуск 1.**

****

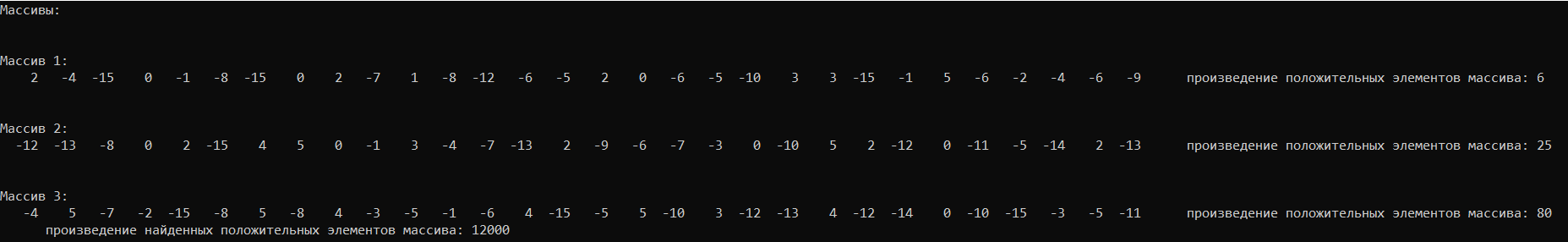
Результат при первом запуске

**Запуск 2.**

****

Результат при втором запуске

**Запуск 3.**

****

Результат при третьем запуске

**Задание 6.** Дано 3 квадратные матрицы А, В и С. Вычислить квадрат той матрицы (А\*А), в которой след наименьший. Вычисление следа оформить в виде функции. Вычисление квадрата матрицы оформить в виде процедуры (тип void). (След матрицы – это сумма элементов главной диагонали)

#include <iostream>

#include <iomanip>

#include <cstdlib>

#include <ctime>

using namespace std;

const int r = 5;

const int col = 5;

void aboba(int a[][col], int r) {

for (int i{}; i < r; i++) {

for (int j{}; j < col; j++) {

a[i][j] = rand() % 21 - 15;

}

}

}

void bubus(int a[][col], int r) {

for (int i{}; i < r; i++) {

for (int j{}; j < col; j++) {

cout << setw(5) << a[i][j];

}

cout << endl;

}

}

int slap(int mat[][col], int net) {

int t = 0;

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

t += mat[i][i];

}

return t;

}

void shiiish(int mata[][col], int res[][col], int net) {

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

for (int j = 0; j < net; ++j) {

res[i][j] = 0;

for (int k = 0; k < net; ++k) {

res[i][j] += mata[i][k] \* mata[k][j];

}

}

}

}

int main() {

setlocale(0, "");

srand(time(0));

int lol[r][col], lel[r][col], heh [r][col], res [r][col];

aboba(lol, r);

cout << "матрица 1:" << endl;

bubus(lol, r);

cout << endl;

aboba(lel, r);

cout << "матрица 2:" << endl;

bubus(lel, r);

cout << endl;

aboba(heh, r);

cout << "матрица 3:" << endl;

bubus(heh, r);

int pop, pep, pip, min=0;

pop = slap(lol, r);

cout << "Сумма главной диагонали первой матрицы: " << pop << endl;

cout << endl;

pep = slap(lel, r);

cout << "Сумма главной диагонали второй матрицы: " << pep << endl;

cout << endl;

pip = slap(heh, r);

cout << "Сумма главной диагонали третьей матрицы: " << pip << endl;

cout << endl;

if (pop < pep && pop < pip) {

shiiish(lol, res, r);

cout << "умножения матрицы 1 на саму себя:" << endl;

bubus(res, r);

cout << endl;

}

else if (pep < pop && pep < pip) {

shiiish(lel, res, r);

cout << "умножения матрицы 2 на саму себя:" << endl;

bubus(res, r);

cout << endl;

}

else if (pip < pop && pip < pep){

shiiish(heh, res, r);

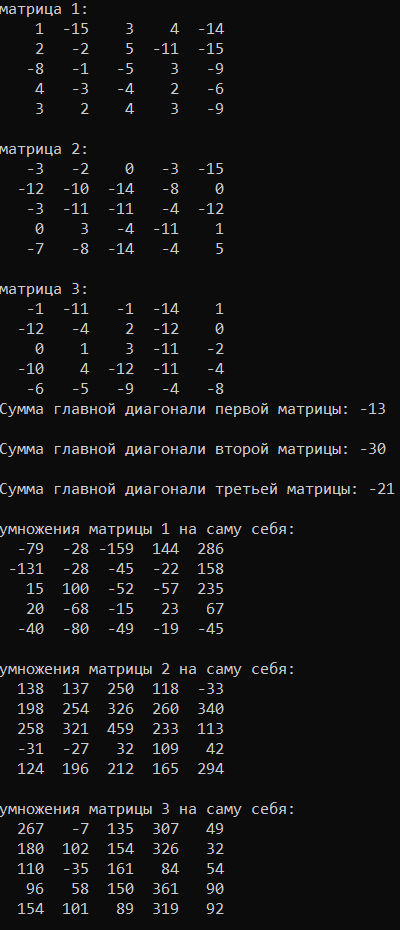
cout << "умножения матрицы 3 на саму себя:" << endl;

bubus(res, r);

}

}

**Запуск 1.**

****

Результат при первом запуске

Изображение выглядит как текст, Шрифт, число, рукописный текст

Контент, сгенерированный ИИ, может содержать ошибки.

Проверка первой матрицы

Изображение выглядит как текст, Шрифт, снимок экрана, число

Контент, сгенерированный ИИ, может содержать ошибки.

Проверка второй матрицы

Изображение выглядит как текст, Шрифт, снимок экрана, число

Контент, сгенерированный ИИ, может содержать ошибки.

Проверка третьей матрицы

**Запуск 2.**

**Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.**

Результат при втором запуске

**Запуск 3.**

**Изображение выглядит как текст, снимок экрана, Шрифт, меню

Контент, сгенерированный ИИ, может содержать ошибки.**

Результат при третьем запуске