**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное автономное образовательное**

**учреждение высшего образования**

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**(РУТ(МИИТ)**

Кафедра «Вычислительные системы, сети и информационная безопасность»

**Отчет По Дисциплине**

**«Алгоритмизация и программирование»**

*Направление:* 10.03.01*Информационная безопасность*

*Профиль:**Безопасность компьютерных систем*

Выполнил:  
студент группы УИБ-112

Орлов Андрей Витальевич

Проверил:

Старший преподаватель Никольская Марина Николаевна

(должность, ФИО)

Старший преподаватель Цыганова Наталия Алексеевна

(должность, ФИО)

Москва 2021 г.

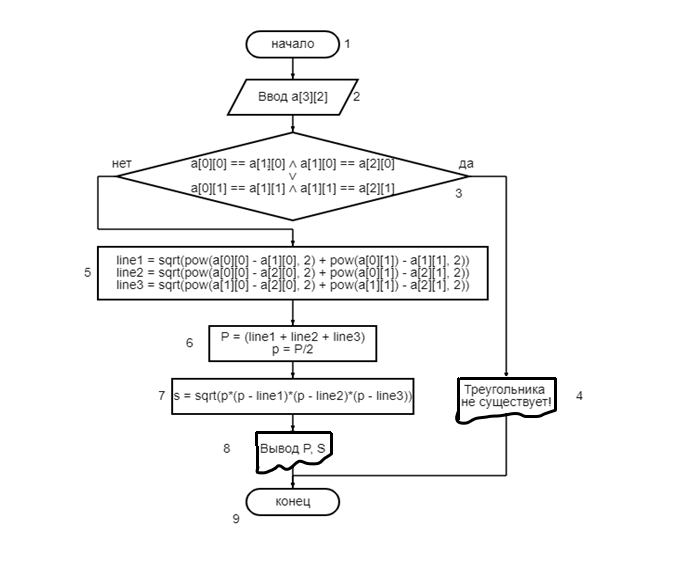
**Задание №1**

Задание: Треугольник задан координатами своих вершин. Найти периметр и площадь треугольника.

1. Таблица имён:

|  |  |  |
| --- | --- | --- |
| Исходные данные | | |
| a[3][2] | вещественный | Массив с координатами |
| Рабочие переменные | | |
| line1 | вещественный | Первая сторона треугольника |
| line2 | вещественный | Вторая сторона треугольника |
| line3 | вещественный | Третья сторона треугольника |
| p | вещественный | Полупериметр треугольника |
| z | символьный | Вывод 'x', 'y' в запросе координат |
| Результат | | |
| P | вещественный | Периметр данного треугольника |
| S | вещественный | Площадь данного треугольника |

2. Блок-схема:



3. Отладочный пример:

Вариант I

1. Начало  
2. Ввод пользователем с клавиатуры массив a[3][2]:  
a[0][0] = 0, a[0][1] = 0, a[1][0] = 6, a[1][1] = 0, a[2][0] = 0, a[2][1] = 8  
3. 0 == 6 == 0 || 0 == 0 == 8 -> нет, идём к блоку 5  
5. Вычисление переменных line1, line2, line3:  
line1 = sqrt((6-0)\*\*2 + (0 – 0)\*\*2) = 6  
line2 = sqrt((0 - 0)\*\*2 + (8 – 0)\*\*2) = 8  
line3 = sqrt((6-0)\*\*2 + (0 – 8)\*\*2) = 10  
6. Вычисление переменных P, p:  
P = 6 + 8 + 10 = 24  
p = 24/2 = 12  
7. Вычисление переменной s:  
s = sqrt(12\*(12 – 10)\*(12 – 6)\*(12 – 8)) = 24  
8. Вывод P, s

9. Конец

Вариант II

1. Начало  
2. Ввод пользователем с клавиатуры a[3][2]:  
a[0][0] = 0, a[0][1] = 1, a[1][0] = 0, a[1][1] = 0, a[2][0] = 0, a[2][1] = -1  
3. 0 == 0 == 0 || 1 == 0 == -1 -> да, идём к блоку 4  
4. Вывод “Треугольника не существует!”  
9. Конец

4. Код программы:

#include <iostream>   
#include <cmath>   
using namespace std;

int main() { // Точка входа в программу  
 setlocale(LC\_ALL, "rus"); // Подключение русского языка  
 double a[3][2], line1, line2, line3, s, p;  
 char z;  
 for (int i = 0; i < 3; i++) { // Заполнение координат  
 cout << "Введите координаты точки " << i + 1 << endl;   
 for (int k = 0; k < 2; k++) {

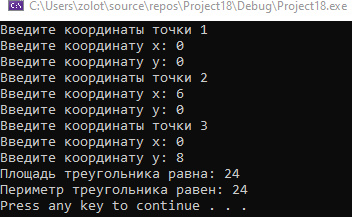
if (k == 0) z = 'x';  
 else z = 'y';  
 cout << "Введите координату " << z << ": ";  
 cin >> a[i][k]; // Ввод координат x, y  
 }  
 }

// Проверка – задали ли нам линию  
 if (a[0][0] == a[1][0] && a[1][0] == a[2][0] || a[0][1] == a[1][1] && a[1][1] == a[2][1]) {  
 cout << "Треугольника не существует!" << endl;  
 return 0;  
}

// расчёт сторон треугольника путём вычисления длин векторов  
 line1 = sqrt(pow(a[0][0] - a[1][0], 2) + pow(a[0][1] - a[1][1], 2));  
 line2 = sqrt(pow(a[0][0] - a[2][0], 2) + pow(a[0][1] - a[2][1], 2));  
 line3 = sqrt(pow(a[1][0] - a[2][0], 2) + pow(a[1][1] - a[2][1], 2));  
 p = (line1 + line2 + line3) / 2;  
// По формуле Герона вычисляем площадь  
 s = sqrt(p \* (p - line1) \* (p - line2) \* (p - line3));   
 cout << "Площадь треугольника равна: " << s << endl;   
 cout << "Периметр треугольника равен: " << p \* 2 << endl;  
 system("pause");  
 return 0;

}

5. Результат выполнения работы программы:

Text

Description automatically generated

Отладка №1 Отладка №2

6. Вывод:

В ходе выполнения работы были изучены математические способы нахождения длин векторов по заданным координатам, а также их реализация с помощью языка высокого уровня.

Было проделано тщательное изучение типов переменных, в ходе которых  
был выбран «double» для оптимальной точности расчёта результата.

На контрольных примерах мы убедились, что код работает корректно.  
Был оформлен комплект документации на программный код.

**Задание №2**

Задание: Вычислить y = cos(x) + cos(x^2) + … + cos(x^n).

1. Таблица имён:

|  |  |  |
| --- | --- | --- |
| Исходные данные | | |
| x | Вещественный | Число – радиант для извлечения косинуса |
| n | целочисленный | Количество слагаемых |
| Рабочие переменные | | |
| I | целочисленный | Счётчик в цикле вычисления суммы |
| c | Вещественный | Увеличитель степени переменной х |
| Результат | | |
| sum | Вещественный | Сумма всех значений косинусов |

2. Блок-схема:

# C:\Users\zolot\OneDrive\Desktop\homework\Алгоритмизация\Прогресс по заданиям\Задание 1 (Треугольник) +\Блок-схема.png

3. Отладочный пример:

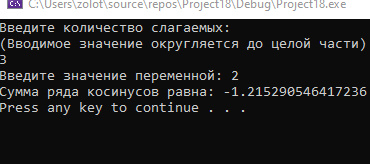
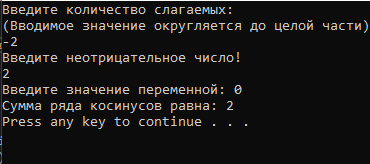
Вариант I  
1. Начало  
2. Ввод чисел с клавиатуры пользователем  
n = -2, x = 0  
4. -2 > 0 нет, идём к блоку 3  
3. Вывод Введите неотрицательное число!  
2. Ввод чисел с клавиатуры пользователя  
n = 2, x = 0  
4. 2 > 0, да, идём к блоку 5  
5. Присваивание переменных  
i = 1, sum = 0, c = x  
6. 1 < 3, да, идём к блоку 7  
7. Переприсваивание переменных  
sum += cos(0) = 0 + 1 = 1, x = x\*c = 0\*0 = 0, i += 1 = 1 + 1 = 2  
6. 2 < 3, да, идём к блоку 7  
7. Переприсваивание переменных  
sum += cos(0) = 1 + 1 = 2, x = x\*c = 0\*0 = 0, i += 1 = 2 + 1 = 3  
6. 3 < 3, нет, идём к блоку 8  
8. Вывод sum = 2 на экран  
9. Конец  
  
Вариант II  
1. Начало  
2. Ввод чисел с клавиатуры пользователем  
n = 3, x = 2  
4. 3 > 0, да, идём к блоку 5  
5. Присваивание переменных  
i = 1, sum = 0, c = x = 2  
6. 1 < 4 да, идём к блоку 7  
7. Переприсваивание переменных  
sum += cos(2) = 0 -0.41614683654714 = -0.41614683654714, x = x\*c = 2\*2 = 4, i += 1 = 1 + 1 = 2  
6. 2 < 4, да, идём к блоку 7  
7. Переприсваивание переменных  
sum += cos(4) = -0.41614683654714 -0.65364362086361 = -1.06979045 741075, x = x\*c = 4\*2 = 8, i += 1 = 2 + 1 = 3  
6. 3 < 4, да, идём к блоку 7  
7. Переприсваивание переменных  
sum += cos(8) = -1.06979045741075 – -0.14550003380861 = -1.215290546417236, x = x\*c = 8\*2 = 16, I += 1 = 3 + 1 = 4  
6. 4 < 4, нет, идём к блоку 8  
8. Вывод sum = 1.215290546417236 на экран  
9. Конец

4. Код программы:

#include <iostream>  
#include <iomanip> // Подключение библиотеки для работы с   
using namespace std;

int main() {  
 float sum = 0, x, c, n;   
 setlocale(LC\_ALL, "rus"); // Подключение русского языка  
 cout << "Введите количество слагаемых:" << endl;  
 cout << "(Вводимое значение округляется до целой части)" << endl;  
 cin >> n;  
 while (n < 0) {  
 cout << "Введите неотрицательное число!" << endl;  
 cin >> n;  
 }  
 cout << "Введите значение переменной: ";  
 cin >> x;  
 c = x;  
 for (int i = 1; i < n + 1; i++) { /\*Вычисление суммы ряда косинусов\*/  
 sum += cos(x);  
 x \*= c;  
 }  
 cout << setprecision(16) <<"Сумма ряда косинусов равна: " << sum << endl;  
 system("pause");  
 return 0;  
}

5. Результат выполнения работы программы:



Отладка №1 Отладка №2

6. Вывод:

В ходе выполнения работы были изучен способ инициализации тригонометрической операции через язык высокого уровня, а также сущность вычисления последовательности через циклы с заданной точностью.

Было проделан анализ кода для очевидной защиты от дурака, поскольку количество слагаемых не может быть отрицательным.

На контрольных примерах мы убедились, что программный код работает корректно при введённых числовых значениях.

Был оформлен комплект документации на программный код.

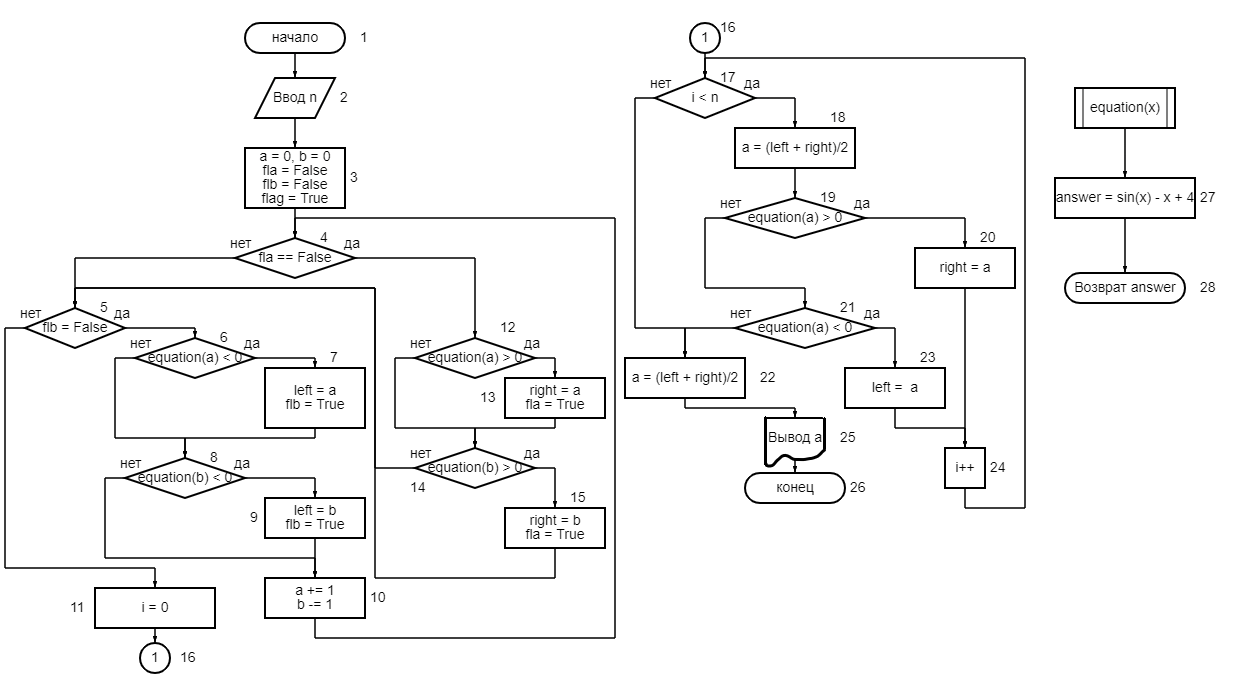
**Задание №3**

Задание 3: вычислить корни уравнения sin(x) – x + 2 = 0 методом деления отрезка пополам.

1. Таблица имён:

|  |  |  |
| --- | --- | --- |
| Исходные данные | | |
| n | целочисленный | Переменная точности вычисления ответа, именно столько раз отрезок ответа будет приближен к решению уравнения |
| Рабочие переменные | | |
| a | вещественный | Изначально переменная с увеличивающимся положительным значением, используется для поиска первейшего отрицательного или положительного значения уравнения, далее используется для подбора более точного нулевого значения уравнения |
| b | целочисленный | Изначально переменная с увеличивающимся отрицательным значением, используется для поиска первейшего отрицательного или положительного значения уравнения |
| fla | булевый | Флаг, сигнализирующий о факте нахождения отрицательного значения функции |
| flb | булевый | Флаг, сигнализирующий о факте нахождения положительного значения функции |
| left | вещественный | Левая граница отрезка ответа |
| right | вещественный | Правая граница отрезка ответа |
| Результат | | |
| a | вещественный | Приближённый ответ уравнения |

2. Блок-схема:



3. Отладочный пример:

Вариант I

1. Начало  
2. Ввод с клавиатуры n = 4  
3.Объявление переменных  
а = 0, b = 0, fla = False, flb = False, flag = True  
4. fla == False, да, идём к блоку 12  
12. Идём в блок 27 для вычисления подпрограммы  
 27. answer = sin(0) – 0 + 4 = 4  
 28. Возврат 4, конец подпрограммы  
4 > 0, да, идём в блок 13  
13.Переприсваивание переменных right = 0, fla = True  
14. Идём в блок 27 для вычисления подпрограммы  
 27. answer = sin(0) – 0 + 4 = 4  
 28. Возврат 4, конец подпрограммы  
4 > 0, да, идём в блок 15  
15. right = 0, fla = True, идём в блок 5  
5. flb == False, да, переход в блок 6  
6. Идём в блок 27 для вычисления подпрограммы  
 27. answer = sin(0) – 0 + 4 = 4  
 28. Возврат 4, конец подпрограммы  
4 < 0, нет, идём в блок 8  
8. Идём в блок 27 для вычисления подпрограммы

27. answer = sin(0) – 0 + 4 = 4  
 28. Возврат 4, конец подпрограммы

4 < 0, нет, идём в блок 10  
10. Переприсваивание переменных  
а = а + 1 = 0 + 1 = 1, b = b = b – 1 = 0 – 1 = -1.  
4. fla == False, нет, идём в блок 5.  
5. flb == False, да, идём в блок 6.

6. Идём в блок 27 для вычисления подпрограммы

27. answer = sin(1) – 1 + 4 = 0.84 + 3 = 3.84  
 28. Возврат 3.84, конец подпрограммы

3, 84 < 0, нет, идём в блок 8.  
8. Идём в блок 27 для вычисления подпрограммы

27. answer = sin(-1) + 1 + 4 = -0.84 + 5 = 4.16  
 28. Возврат 4.16, конец подпрограммы

4.16 < 0, нет, идём в блок 10.  
10. Переприсваивание переменных  
a = a + 1 = 1 + 1 = 2, b = b – 1, b = -1 -1 = -2.  
4. fla == False, нет, идём в блок 5  
5. flb == False, да, идём в блок 6

6. Идём в блок 27 для вычисления подпрограммы

27. answer = sin(2) – 2 + 4 = 0.9 + 2 = 2.9  
 28. Возврат 2.9, конец подпрограммы  
2.9 < 0, нет, идём в блок номер 8.  
8. Идём в блок 27 для вычисления подпрограммы

27. answer = sin(-2) + 2 + 4 = -0.9 + 6 = 5.1  
 28. Возврат 5.1, конец подпрограммы

5.1 < 0, нет, идём в блок номер 10  
10. Переприсваивание переменных  
a = a + 1 = 2 + 1 = 3, b = b – 1 = - 2 - 1 = -3

4. fla == False, нет, идём в блок 5  
5. flb == False, да, идём в блок 6.

6. Идём в блок 27 для вычисления подпрограммы

27. answer = sin(3) – 3 + 4 = 0.14 + 1 = 1.14  
 28. Возврат 1.14, конец подпрограммы  
1.14 < 0, нет, идём в блок номер 8.  
8. Идём в блок 27 для вычисления подпрограммы

27. answer = sin(-3) + 3 + 4 = -0.14 + 7 = 6.86  
 28. Возврат 6.86, конец подпрограммы

6.86 < 0, нет, идём в блок номер 10

10. Переприсваивание переменных  
a = a + 1 = 3 + 1 = 4, b = b – 1 = - 3 - 1 = - 4

4. fla == False, нет, идём в блок 5  
5. flb == False, да, идём в блок 6.   
6. Идём в блок 27 для вычисления подпрограммы

27. answer = sin(4) – 4 + 4 = -0.75  
 28. Возврат -0.75, конец подпрограммы  
-0.75 < 0, да, идём в блок номер 7.

7. left = a = 4, flb = True  
8. Идём в блок 27 для вычисления подпрограммы

27. answer = sin(-4) + 4 + 4 = 0.75 + 7 = 7.75  
 28. Возврат 7.75, конец подпрограммы

7.75 < 0, нет, идём в блок номер 10  
10. Переприсваивание переменных  
a = a + 1 = 4 + 1 = 5, b = b – 1 = -4 -1 = -5  
4. fla == False, нет, идём в блок 5.  
5. flb == False, нет, идём в блок 11.  
11. i = 0  
16. Ссылка на блок 17.  
17. 0 < 4, да, идём в блок номер 16.  
16. Переприсваивание переменной  
a = (4 + 0)/2 = 2  
19. Идём в блок 27 для вычисления подпрограммы  
 27. answer = sin(2) – 2 + 4 = 0.9 + 2 = 2.9  
 28. Вывод 2.9  
 29. Возврат 2.9, конец подпрограммы  
2.9 > 0, да, идём к блоку 20  
20. Переприсваивание переменных  
right = a = 2  
24. i = i + 1 = 0 + 1 = 1  
17. 1 < 4, да, идём в блоку 18  
18. a = (4 + 2)/2 = 3  
19. Идём в блок 27 для вычисления подпрограммы  
 27. answer = sin(3) – 3 + 4 = 0.14 + 1 = 1.14  
 28. Возврат 1.14, конец подпрограммы  
1.14 > 0, да, идём к блоку 20.  
20. Переприсваивание переменных.   
right = 3  
24. i = i + 1 = 1 + 1 = 2  
17. 2 < 4, да, идём к блоку 18  
18. a = (4 + 3)/2 = 3.5  
19. Идём в блок 27 для вычисления подпрограммы  
 27. answer = sin(3.5) – 3.5 + 4 = -0.35 + 0.5 = 0.15  
 28. Возврат 0.15, конец подпрограммы  
0.15 > 0, да, идём к блоку 20.  
20. Переприсваивание переменных  
right = a = 3.5  
24. i = i + 1 = 2 + 1 = 3  
17. 3 < 4, да, идём к блоку 18  
18. a = (4 + 3.5)/2 = 3.75

19. Идём в блок 27 для вычисления подпрограммы  
 27. answer = sin(3.75) – 3.75 + 4 = -0.57 + 0.25 = -0.32  
 28. Возврат -0.32, конец подпрограммы  
-0.32 > 0, нет, идём к блоку 21.  
21. Идём в блок 27 для вычисления подпрограммы  
 27. answer = sin(3.75) – 3.75 + 4 = -0.57 + 0.25 = -0.32  
 28. Возврат -0.32, конец подпрограммы  
-0.32 < 0, да, идём в блок 23  
23. Переприсваивание переменных  
left = a = 3.75  
24. I = i + 1 = 3 + 1 = 4  
17. 4 < 4, нет, идём в блок 22  
22. a = (3.75 + 3.5)/2 = (7.25)/2 = 3.625  
25. Вывод a  
26. Конец

4. Код программы:

#include <iostream>

#include <cmath>

#include <iomanip> // Подключение библиотеки для манипуляций вводом\выводом, в частности - setprecision().

using namespace std;

float equation(float x) {

return double(sin(x) - x + 4);

}

int main() {

setlocale(LC\_ALL, "Russian");

// Объявление переменных точностей  
 int n, accuracy;

cout << "Введите точность вычислений корня уравнения: ";

cin >> n;

cout << "Введите количество знаков после запятой (Незначащие нули не будут выведены): ";

cin >> accuracy;

double a = 0, b = 0, right, left;

// Объявление флагов нахождения границ отрезка функции в формате [-x; x], x > 0

bool flag = true, fla = false, flb = false;

while (flag) {

if (!fla) { // Пока не найдена правая граница отрезка

if (equation(a) > 0) {

right = a; // правая граница отрезка найдена

fla = true; // меняем булевое значение флага

}

if (equation(b) > 0) {

right = b; // правая граница отрезка найдена,

fla = true; // меняем булевое значение флага

}

}

if (!flb) { // Пока не найдена левая граница отрезка

if (equation(a) < 0) {

left = a; // левая граница отрезка найдена,

flb = true; // меняем булевое значение флага

}

if (equation(b) < 0) {

left = b; // левая граница отрезка найдена,

flb = true; // меняем булевое значение флага, }

}

a += 1;

b -= 1;

if (fla && flb) { // Если нашлись границы отрезка

break; // Выходим из цикла

}

}

// Инициализация цикла вычисления корня уравнения

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

// Переменная а используется для краткости программы, в нее записываем середину действующего отрезка

a = (left + right) / 2;

// Если значение функции меньше нуля, тогда середина становится левой границей отрезка

if (equation(a) > 0) {

right = a;

}

// Если значение функции меньше нуля, тогда середина становится левой границей отрезка

else if (equation(a) < 0) {   
 left = a;

}

// Если границы отрезка идентичны друг другу, тогда мы нашли корень, выходим из цикла

else {

break;

}

}

a = (left + right) / 2;

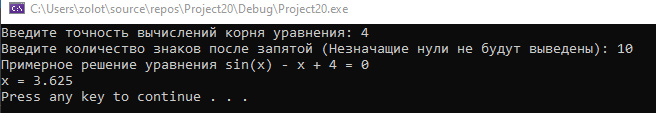
cout << "Примерное решение уравнения sin(x) - x + 4 = 0\nx = " << setprecision(accuracy) << a << endl;

system("pause");

return 0;

}

5. Результат выполнения работы программы:



Отладочный пример №1

6. Вывод:

В ходе выполнения работы были изучены способы нахождения решения уравнения путём нахождения приблизительного значения ответа методом исследования числовой оси на отрезок с границами, в которых функция различается по знаку. Далее с помощью деления отрезка пополам запрашиваемое количество раз находим приблизительный ответ.

Было проделано несколько опытов на других уравнениях, где такой способ вполне сработал, так что алгоритм универсален.

На контрольных примерах мы убедились, что алгоритм работает верно.

Был оформлен комплект документации на программный код.

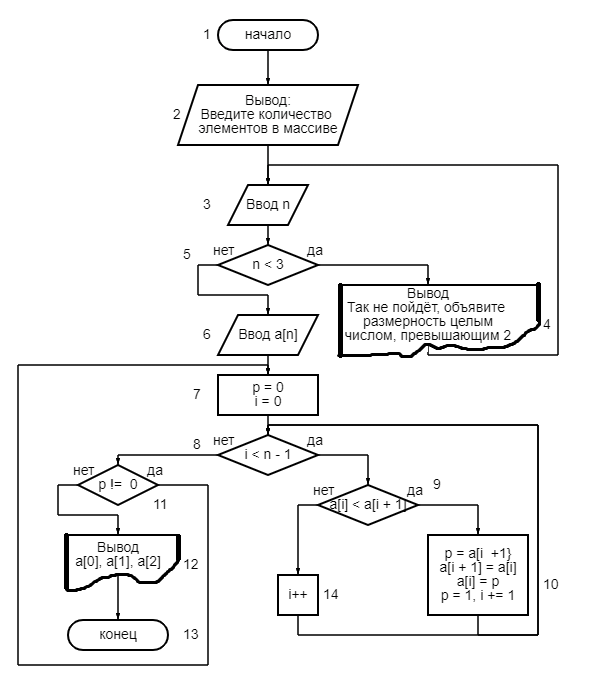
**Задание №4**

Задание: Задан одномерный массив целых чисел. Определить 3 наибольших элементов в нём.

1. Таблица имён:

|  |  |  |
| --- | --- | --- |
| Исходные данные | | |
| n | целочисленный | Количество элементов в массиве |
| a[n] | целочисленный | Массив элементов размерности n |
| Рабочие переменные | | |
| p | целочисленный | Вспомогательная переменная при реализации обмена значений переменных, флаг наличия перестановки элементов |
| i | целочисленный | Переменная – счётчик в цикле |
| Результат | | |
| a[0] | целочисленный | Наибольший элемент в массиве |
| a[1] | целочисленный | Второй наибольший элемент в массиве |
| a[2] | целочисленный | Третий наибольший элемент в массиве |

2. Блок-схема:



3. Отладочный пример:

Вариант I

1. Начало  
2. Вывод Введите количество элементов в массиве  
3. Ввод переменной пользователем  
n = 2  
5. 2 < 3, да, идём к блоку 4.  
4. Вывод Так не пойдёт, объявите размерность целым числом  
3. Ввод переменной пользователем  
n = 4  
5. 4 < 3, нет, идём к блоку 6  
6. Вводим элементы массива a[0] = 1, a[1] = 2, a[2] = 4, a[3] = 3  
7. Присваивание значений переменных  
p = 0, I = 0  
8. 0 < 3, да, идём к блоку 9  
9. 1 < 2, да, идём к блоку 10  
10. Переприсваивание переменных  
p = 2, a[1] = 1, a[0] = 2, p = 1, i += 1 = 0 + 1 = 1  
8. 1 < 3, да, идём к блоку 9  
9. 1 < 4, да, идём к блоку 10  
10. Переприсваивание переменных  
p = 4, a[2] = 1, a[1] = 4, p = 1, i += 1 + 1 = 2  
8. 2 < 3, да, идём к блоку 9  
9. 1 < 3, да, идём к блоку 10  
10. Переприсваивание переменных  
p = 3, a[3] = 1, a[2] = 3, p = 1, i += 1 = 2 + 1 = 3  
8. 3 < 3, нет, идём к блоку номер 11  
9. 1 != 0, да, идём к блоку 7  
7. Переприсваивание переменных  
p = 0, I = 0  
8. 0 < 3, да, идём к блоку 9  
9. 2 < 4, да, идём к блоку 10  
10. Переприсваивание переменных  
p = 4, a[1] = 2, a[0] = 4, p = 1, i += 1 = 0 + 1 = 1  
8. 1 < 3, да, идём к блоку 9  
9. 2 < 3, да, идём к блоку 10  
10. Переприсваивание переменных  
p = 3, a[2] = 2, a[1] = 3, p = 1, i += 1 + 1 = 2  
8. 2 < 3, да, идём к блоку 9  
9. 2 < 1, нет, идём к блоку 14  
14. i += 1 = 2 + 1 = 3  
8. 3 < 3, нет, идём к блоку 11  
11. 1 != 0, да, идём к блоку 7  
7. Переприсваивание переменных  
p = 0, i = 0  
8. 0 < 3, да, идём к блоку 9  
9. 4 < 3, нет, идём в блок 14  
14. i += 1 = 0 + 1 = 1  
8. 1 < 3, да, идём в блок 9  
9. 3 < 2, нет, идём в блок 14  
14. i += 1 = 1 + 1 = 2  
8. 2 < 3, да, идём в блок 9  
9. 2 < 1, нет, идём в блок 14  
14. i += 1 = 2 + 1 = 3  
8. 3 < 3, нет, идём в блок 11  
11. 0 != 0, нет, идём в блок 12  
12. Вывод 4, 3, 2  
13. Конец

4. Код программы:

#include <iostream>   
using namespace std;   
int main() { // Точка входа в программу

setlocale(LC\_ALL, "Russian"); // Подключение русского языка

int n, p, \*a;

cout << "Введите количество элементов в массиве: ";  
 cin >> n;

while (n < 3) { // Проверка на размерность

if (n < 3) {

cout << "Так не пойдёт, объявите размерность целым числом, превышающим 2";

}

　　　　　сin >> n;

}

a = new int[n]; // Объявление массива размерности n

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

for (int i = 0; i < n; i++) { // Цикл ввода элементов массива

cin >> a[i];

}

do { // Метод сортировки пузырьком по убыванию

p = 0;

for (int i = 0; i < n - 1; i++) {

if (a[i] < a[i + 1]) {

p = a[i + 1];

a[i + 1] = a[i];

a[i] = p;

p = 1;

}

}

} while (p != 0);

cout << "3 наибольших числа: "; // Непосредственно вывод ответа

cout << a[0] << " ";

cout << a[1] << " ";

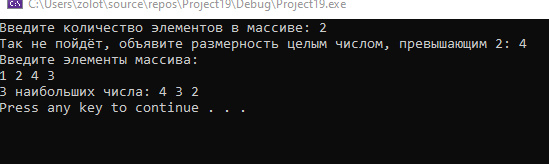
cout << a[2] << "\n";

system("pause"); // Консоль ожидает действий пользователя

return 0;

}

5. Результат выполнения работы программы:



Отладка №1

6. Вывод:

В ходе выполнения работы были изучены способы использования массивов на языке высокого уровня, а также сортировка пузырьком по убыванию, что позволило упростить задачу до вывода первых трёх элементов.

Была проделана небольшая работа для красивого оформления программы в формате “User-friendly”, чтобы пользователь мог ориентироваться в ней.

На контрольных примерах мы убедились, что сортировка работает корректно, а массив произвольного размера задаётся вполне конкретно.

Был оформлен комплект документации на программный код.

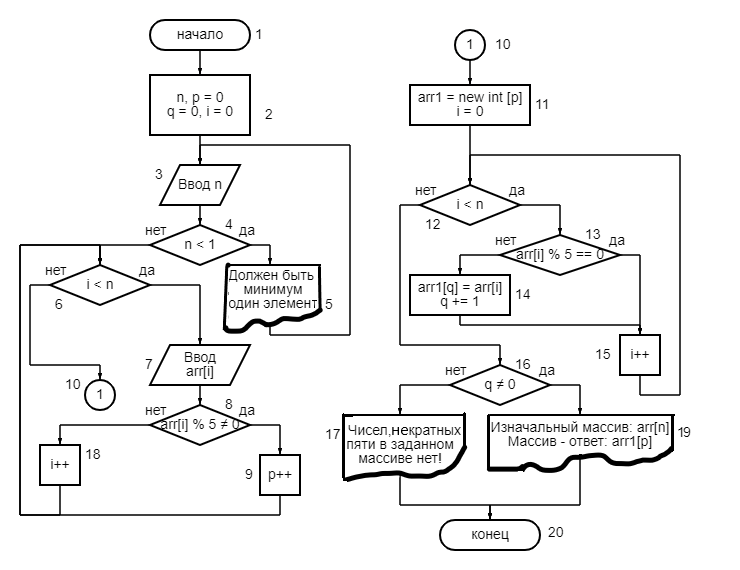
**Задание №5**

Задание: Дан массив целых чисел. Удалить из заданного массива все элементы, кратные 5.

1. Таблица имён:

|  |  |  |
| --- | --- | --- |
| Исходные данные | | |
| n | целочисленный | Размерность изначального массива |
| arr[n] | целочисленный | Массив целых чисел |
| Рабочие переменные | | |
| p | целочисленный | Количество элементов, не кратных пяти |
| q | целочисленный | Переменная-счётчик для инициализации массива без чисел, кратных пяти |
| i | целочисленный | Переменная-счётчик в циклах |
| Результат | | |
| arr1[p] | целочисленный | Массив без чисел, кратных пяти |

2. Блок-схема:



3. Отладочный пример:

Вариант I

1. Начало  
2. Инициализация переменных  
p = 0, q = 0, i = 0  
3. Ввод n = 0  
4 0 < 1, да, идём к блоку 5  
5. Вывод Должен быть минимум один элемент  
3. Ввод n = 3  
4. 3 < 1, нет, идём к блоку 6  
6. 0 < 3, да ,идём к блоку 7  
7. Ввод arr[0] = 5   
8. 5 % 5 != 0, нет, идём к блоку 18  
18. i++ = 0 + 1 = 1  
6. 1 < 3, да, идём к блоку 7  
7. Ввод arr[1] = 7  
8. 7 % 5 != 0, да, идём в блок 9  
9. p++ = 0 + 1 = 1  
18. i++ = 1 + 1 = 2  
6. 2 < 3, да, идём к блоку 7  
7. Ввод arr[2] = 14  
8. 14 % 5 != 0, да, идём к блоку 9  
9. p++ = 1 + 1 = 2  
18. i++ = 2 + 1 = 3  
6. 3 < 3, нет, идём в ссылку 10  
10. Идём в блок 11  
11. Переприсваивание переменных  
arr1 = new int [2], i = 0  
12. 0 < 3, да, идём в блок 13  
13. 5 % 5 == 0, да, идём в блок 15  
15. i++ = 0 + 1 = 1  
12. 1 < 3, да, идём в блок 13  
13. 7 % 5 == 0, нет, идём в блок 14  
14. arr1[0] = 7, q++ = 0 + 1 = 1  
15. i++ = 1 + 1 = 2  
12. 2 < 3, да, идём в блок 13  
13. 14 % 5 == 0, нет, идём в блок 14  
14. arr1[1] = 14, q++ = 1 + 1 = 2  
15. i++ = 2 + 1 = 3  
12. 3 < 3, нет. Идём к блоку 16  
16. 2 != 0, да, идём в блок 19  
19. Вывод   
Изначальный массив: 5 7 14   
Изменённый массив: 7 14  
20. Конец  
  
 Вариант II

1. Начало  
2. Инициализация переменных p = 0, q = 0, i = 0  
3. Ввод n = 1  
4. 1 < 1, нет, идём в блок 6  
6. 0 < 1, да, идём в блок 7  
7. Ввод arr[0] = 5  
8. 5 % 5 != 0, нет, идём в блок 18  
18. i += 1 = 0 + 1 = 1  
6. 1 < 1, нет, идёс в блок 10  
10. Ссылка на блок 11  
11. Переприсваивание переменных arr1 = new int[p], i = 0  
12. 0 < 1, да, идём в блок 13.  
13. arr[0] % 5 == 0 -> 5 % 5 == 0, да, идём в блок 15  
15. i += 1 = 0 + 1 = 1  
12. 1 < 1, нет, идём в блок 16  
16. 0 != 0, нет, идём в блок 17  
17. Вывод Чисел, некратных пяти в заданном массиве нет!  
20. Конец

4. Код программы:

#include <iostream>

using namespace std;

int main() { //Точка входа в программу

setlocale(LC\_ALL, "Russian");

int n, p = 0, \* arr, \* arr1, q = 0;

cout << "Введите количество элементов массива: ";

cin >> n;

while (n < 1) {

if (n < 1) {

cout << "Не пойдёт, должен быть как минимум 1 элемент: ";

}

cin >> n;

}

cout << "Введите элементы массива: ";

arr = new int[n]; // Объявление массива arr размерности n.

//Заполнение массива и подсчёт элементов, не кратных пяти

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

cin >> arr[i];

if (arr[i] % 5 != 0) {

p++;

}

}

arr1 = new int[p]; // Объявление массива arr1 размерности p

// Заполняем arr1, выискивая элементы не кратные 5 в изначальном массиве.

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

if (arr[i] % 5 == 0) {

continue;

}

arr1[q] = arr[i];

q += 1;

}

// Вывод изначального массива и изменнного.

if (q != 0) {

cout << "Изначальный массив: ";

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

cout << arr[i] << " ";

}

cout << endl << "Массив - ответ: ";

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

cout << arr1[i] << " ";

}

}

else

cout << "Чисел, не кратных пяти, в заданном массиве нет!" << endl;

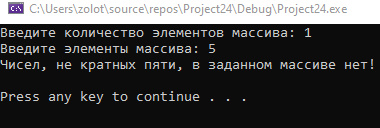
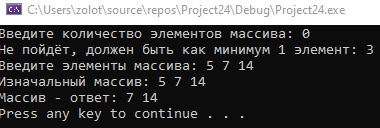
cout << endl;

system("pause");

return 0;

}

5. Результат выполнения работы программы:



Отладочный пример №1 Отладочный пример №2

6. Вывод:

В ходе выполнения работы были изучены методы заполнения массивов, а также поиск определённых элементов в них при помощи условного оператора и цикла, перебирающего значения массива через переменную-счётчик.

Было проделано

На контрольных примерах мы убедились, что исходный код работает корректно во всех точках программы, включая точки обработки исключения (Когда размерность, заданная юзером, не была бы натуральным числом)

Был оформлен комплект документации на программный код.

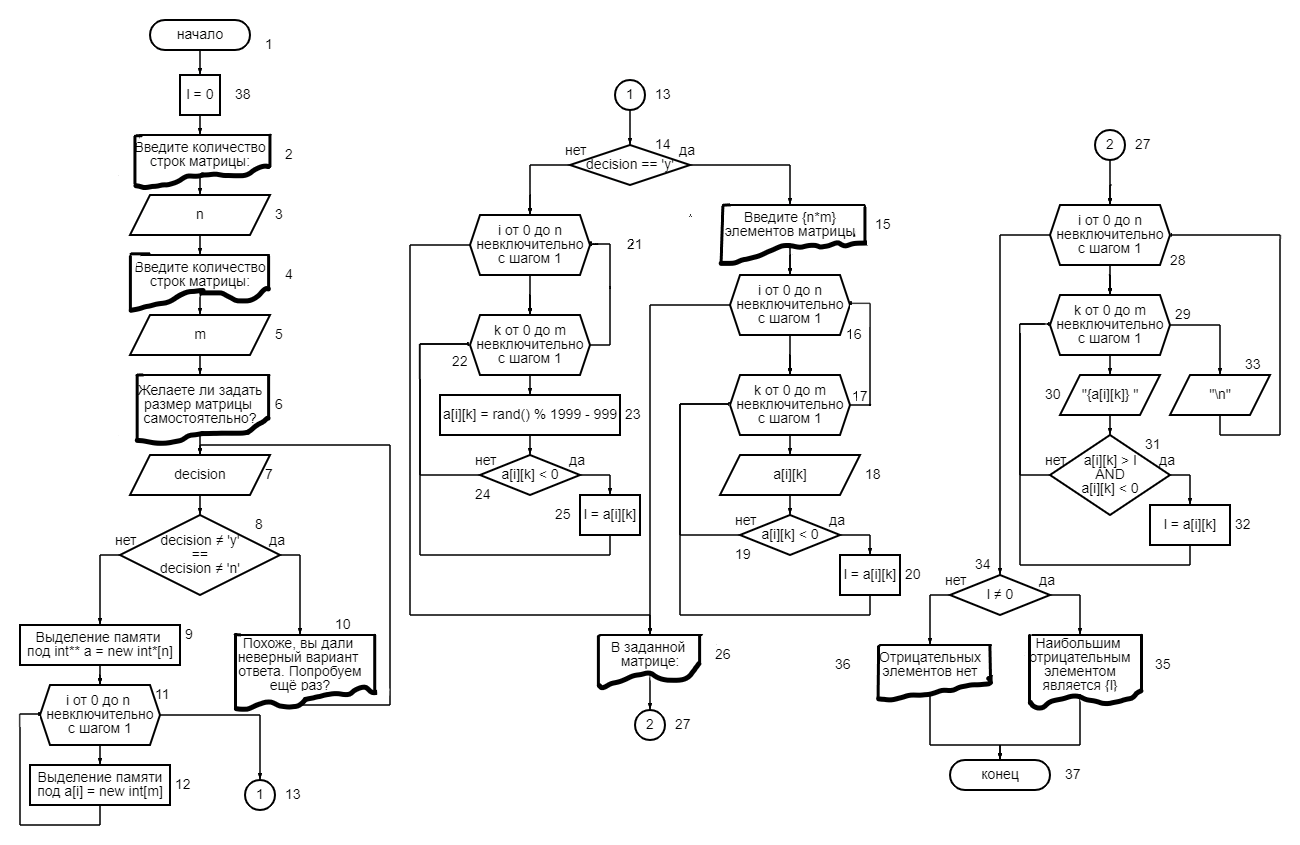
**Задание №6**

Задание: Задана матрица целых чисел. Определить максимальный элемент среди отрицательных элементов этой матрицы.

1. Таблица имён:

|  |  |  |
| --- | --- | --- |
| Исходные данные | | |
| n | целочисленный | Количество строк в матрице |
| m | целочисленный | Количество столбцов в матрице |
| a[n][m] | целочисленный | Массив, содержащий исходную матрицу |
| decision | символьный | Определитель типа заполнение матрицы |
| Рабочие переменные | | |
| i | целочисленный | Переменная-счётчик в циклах |
| k | целочисленный | Переменная-счётчик в циклах |
| a[i][k] | целочисленный | Элемент матрицы i строки, j столбца |
| Результат | | |
| l | целочисленный | Наибольший отрицательный элемент |

2. Блок-схема:



3. Отладочный пример:

Вариант I

1. Начало  
38. Присваивание переменной l = 0  
2. Вывод «Введите количество строк матрицы: »  
3. Ввод n = 3.  
4. Вывод «Введите количество столбцов матрицы: »  
5. Ввод m = 3.  
6. Вывод «Желаете ли задать размер матрицы самостоятельно?»  
7. Ввод decision = “Bone”  
8. “Bone” != ‘y’ == “Bone” != ‘n’, да, переход к блоку 10.  
10. Вывод «Похоже, вы дали неверный вариант ответа. Попробуем ещё раз?»  
7. Ввод decision = 'y'.  
8. “y” != ‘y’ == “y” != ‘n’, нет, переход к блоку 9.  
9. Выделение памяти под массив массивов a размерности 3.  
11. Вход в цикл: Переменная i = 0.  
12. Выделение памяти под элемент массива массивов a[0] размерности 3.  
11. Вход в цикл: Переменная i = 1.  
12. Выделение памяти под элемент массива массивов a[1] размерности 3.  
11. Вход в цикл: Переменная i = 2.  
12. Выделение памяти под элемент массива массивов a[2] размерности 3.  
11. Вход в цикл: Переменная i = 3. Граница достигнута, переход в блок 13.  
13. Ссылка на блок 14.  
14. 'y' == 'y', да, идём в блок 15.  
15. Вывод «Введите 9 элементов матрицы»  
16. Вход в цикл: Переменная i = 0.  
17. Вход в цикл: Переменная k = 0.  
18. Ввод a[0][0] = 1  
19. 1 < 0, нет, идём к блоку 17  
17. Вход в цикл: Переменная k = 1.  
18. Ввод a[0][1] = 7  
19. 7 < 0, нет, идём к блоку 17  
17. Вход в цикл: Переменная k = 2.  
18. Ввод a[0][2] = -4  
19. -4 < 0, да, идём к блоку 20  
20. Переприсваивание переменных l = -4  
17. Вход в цикл: Переменная k = 3. Граница достигнута, переход в блок 16.  
16. Вход в цикл: Переменная i = 1.  
17. Вход в цикл: Переменная k = 0.  
18. Ввод a[1][0] = 2  
19. 2 < 0, нет, идём к блоку 17  
17. Вход в цикл: Переменная k = 1.  
18. Ввод a[1][1] = 14  
19. 14 < 0, нет, идём к блоку 12  
17. Вход в цикл: Переменная k = 2.  
18. Ввод a[1][2] = -8  
19. -8 < 0, да, идём к блоку 11  
20. Переприсваивание переменных l = -8  
17. Вход в цикл: Переменная k = 3. Граница достигнута, переход в блок 16  
16. Вход в цикл: Переменная i = 2.  
17. Вход в цикл: Переменная k = 0.  
18. Ввод a[2][0] = 0  
19. 0 < 0, нет, идём к блоку 17  
17. Вход в цикл: Переменная k = 1.  
18. Ввод a[2][1] = 3  
19. 3 < 0, нет, идём к блоку 17  
17. Вход в цикл: Переменная k = 2.  
18. Ввод a[2][2] = -11  
19. -11 < 0, да, идём к блоку 20  
20. Переприсваивание переменных l = -11  
17. Вход в цикл: Переменная k = 3. Граница достигнута, переход в блок 16  
16. Вход в цикл: Переменная i = 3. Граница достигнута, переход в блок 26  
26. Вывод «В заданной матрице: »  
27. Ссылка на блок 28  
28. Вход в цикл: Переменная i = 0.  
29. Вход в цикл: Переменная k = 0.  
30. Вывод «1 »  
31. 1 > -11 && 1 < 0, нет, идём к блоку 29  
29. Вход в цикл: Переменная k = 1.  
30. Вывод «7 »  
31. 7 > -11 && 7 < 0, нет, идём к блоку 29  
29. Вход в цикл: Переменная k = 2.  
30. Вывод «-4 »  
31. -4 > -11 && -4 < 0, да, идём к блоку 32  
32. Переприсваивание переменных l =-4  
29. Вход в цикл: Переменная k = 3. Граница достигнута, переход в блок 28  
28. Вход в цикл: Переменная i = 0.  
29. Вход в цикл: Переменная k = 0.  
30. Вывод «2 »  
31. 2 > -11 && 2 < 0, нет, идём к блоку 29  
29. Вход в цикл: Переменная k = 1.  
30. Вывод «14 »  
31. 14 > -11 && 14 < 0, нет, идём к блоку 29  
29. Вход в цикл: Переменная k = 2.  
30. Вывод «-8 »  
31. -8 > -11 && -8 < 0, нет, идём к блоку 29  
29. Вход в цикл: Переменная k = 3. Граница достигнута, переход в блок 28  
33. Перенос текста на новую строку.  
28. Вход в цикл: Переменная i = 1.  
29. Вход в цикл: Переменная k = 0.  
30. Вывод «0 »  
31. 0 > -4 && 0 < 0, нет, идём к блоку 29  
29. Вход в цикл: Переменная k = 1.  
30. Вывод «3 »  
31. 3 > -4 && 3 < 0, нет, идём к блоку 29  
29. Вход в цикл: Переменная k = 2.  
30. Вывод «-11 »  
31. -11 > -4 && -11 < 0, нет, идём к блоку 29  
29. Вход в цикл: Переменная k = 3. Граница достигнута, переход в блок 28  
33. Перенос текста на новую строку.  
34. -4 != 0, да, переход к блоку 35  
35. Вывод «Наибольшим отрицательным элементом является -4»  
36. Конец

Вариант II.   
1. Начало  
2. Вывод «Введите количество строк матрицы: »  
3. Ввод n = 2.  
4. Вывод «Введите количество столбцов матрицы: »  
5. Ввод m = 3.  
6. Вывод «Желаете ли задать размер матрицы самостоятельно?»  
7. Ввод decision = 'n'.  
8. “n” != ‘y’ == “n” != ‘n’, нет, переход к блоку 9.  
9. Выделение памяти под массив массивов a размерности 2.  
11. Вход в цикл: Переменная i = 0.  
12. Выделение памяти под элемент массива массивов a[0] размерности 3.  
11. Вход в цикл: Переменная i = 1.  
12. Выделение памяти под элемент массива массивов a[1] размерности 3.  
11. Вход в цикл: Переменная i = 2. Граница достигнута, переход в блок 13.  
13. Ссылка на блок 14.  
14. 'n' == 'y', нет, идём в блок 21.  
21. Вход в цикл: Переменная i = 0.  
22. Вход в цикл: Переменная k = 0.  
23. Выбор числа из промежутка [-999;1999) для a[0][0]: 447  
24. 447 < 0, нет, переход в блок 22  
22. Вход в цикл: Переменная k = 1.  
23. Выбор числа из промежутка [-999;1999) для a[0][1]: 809  
24. 809 < 0, нет, переход в блок 22  
22. Вход в цикл: Переменная k = 2  
23. Выбор числа из промежутка [-999;1999) для a[0][2]: 1495  
24. 1495 < 0, нет, переход в блок 22  
22. Вход в цикл: Переменная k = 3. Граница достигнута, переход в блок 21  
21. Вход в цикл: Переменная i = 1.  
22. Вход в цикл: Переменная k = 0.  
23. Выбор числа из промежутка [-999;1999) для a[1][0]: 184  
24. 184 < 0, нет, переход в блок 22  
22. Вход в цикл: Переменная k = 1.  
23. Выбор числа из промежутка [-999;1999) для a[1][1]: 1724  
24. 1724 < 0, нет, переход в блок 22  
22. Вход в цикл: Переменная k = 2  
23. Выбор числа из промежутка [-999;1999) для a[2][0]: 1016  
24. 1016 < 0, нет, переход в блок 22  
22. Вход в цикл: Переменная k = 3. Граница достигнута, переход в блок 21  
21. Вход в цикл: Переменная k = 2. Граница достигнута, переход в блок 26  
26. Вывод «В заданной матрице »  
27. Ссылка на блок 28  
28. Вход в цикл: Переменная i = 0  
29. Вход в цикл: Переменная k = 0  
30. Вывод «447 »  
31. 447 > 0 && 447 < 0, нет, идём в блок 29  
29. Вход в цикл: Переменная k = 1.  
30. Вывод «809 »  
31. 809 > 0 && 809 < 0, нет, идём в блок 29  
29. Вход в цикл: Переменная k = 2.  
30. Вывод «1495 »  
31. 1495 > 0 && 1495 < 0, нет, идём в блок 29  
29. Вход в цикл: Переменная k = 3. Граница достигнута, переход в блок 28.  
28. Вход в цикл: Переменная i = 1  
29. Вход в цикл: Переменная k = 0  
30. Вывод «184 »  
31. 184 > 0 && 184 < 0, нет, идём в блок 29  
29. Вход в цикл: Переменная k = 1.  
30. Вывод «1724 »  
31. 1724 > 0 && 1724 < 0, нет, идём в блок 29  
29. Вход в цикл: Переменная k = 2.  
30. Вывод «1016»  
31. 1016 > 0 && 1016 < 0, нет, идём в блок 29  
29. Вход в цикл: Переменная k = 3. Граница достигнута, переход в блок 28.  
28. Вход в цикл: Переменная i = 2. Граница достигнута, переход в блок 34.  
34. 0 != 0, нет, переход к блоку 36  
36. Вывод «Отрицательных элементов нет»  
37. Конец  
  
Вариант III.

1. Начало  
2. Вывод «Введите количество строк матрицы: »  
3. Ввод n = 1.  
4. Вывод «Введите количество столбцов матрицы: »  
5. Ввод m = 1.  
6. Вывод «Желаете ли задать размер матрицы самостоятельно?»  
7. Ввод decision = 'n'.  
8. “n” != ‘y’ == “n” != ‘n’, нет, переход к блоку 9.  
9. Выделение памяти под массив массивов a размерности 1.  
11. Вход в цикл: Переменная i = 0.  
12. Выделение памяти под элемент массива массивов a[0] размерности 1.  
11. Вход в цикл: Переменная i = 1. Граница достигнута, переход в блок 13  
13. Ссылка на блок 14  
14. 'n' == 'y', нет, переход в блок 21  
21. Вход в цикл: Переменная i = 0.  
22. Вход в цикл: Переменная k = 0.  
23. Выбор числа из промежутка [-999;1999) для a[1][0]: -988  
24. -988 < 0, да, переход к блоку 25  
25. Переприсваивание переменных l = -988  
22. Вход в цикл: Переменная k = 1. Граница достигнута, переход в блок 21  
21. Вход в цикл: Переменная i = 1. Граница достигнута, переход в блок 26  
26. Вывод «В заданной матрице: »  
27. Ссылка на блок 28  
28. Вход в цикл: Переменная i = 0.  
29. Вход в цикл: Переменная k = 0.  
30. Вывод «-988 »  
31. -988 > l && -988 < 0, нет, переход к блоку 29.  
29. Вход в цикл: Переменная k = 1. Граница достигнута, переход в блок 33  
33. Перенос текста на следующую строчку  
28. Вход в цикл: Переменная i = 1. Граница достигнута, переход в блок 34  
34. -988 != 0, да, переход к блоку 35  
35. Вывод «Наибольшим отрицательным элементом является -988»  
37. Конец  
  
 4. Код программы:

#include <iostream>

#include <ctime>

using namespace std;

int main() {

srand(time(NULL)); // Для различных показаний функции rand()

setlocale(LC\_ALL, "rus");

int n, m, l = 0; char decision;

cout << "Введите количество строк матрицы: ";

cin >> n;

cout << "Введите количество столбцов матрицы: ";

cin >> m;

cout << "Желаете ли задать размер матрицы самостоятельно? (y/n)" << endl;

cin >> decision;

while (decision != 'y' && decision != 'n') {

cout << "Похоже, вы дали неверный вариант ответа. Попробуем ещё раз?\n";

cout << "Желаете ли задать элементы матрицы самостоятельно? (y/n)" << endl;

cin >> decision;

}

int\*\* a = new int\* [n];

// Выделяем память под каждый элемент массива указателей

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

a[i] = new int[m];

}

if (decision == 'y') {

cout << "Введите " << n \* m << " элементов матрицы" << "\n";

//Заполнение двумерного массива матрицей, поиск рандомного отрицательного числа

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

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

cin >> a[i][k];

if (a[i][k] < 0) {

l = a[i][k];

}

}

}

}

else {

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

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

a[i][k] = rand() % 1999 - 999;

if (a[i][k] < 0) {

l = a[i][k];

}

}

}

}

//Вывод двумерного массива, поиск наибольшего отрицательного числа

cout << "В заданной матрице" << endl;

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

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

cout << a[i][k] << " ";

if (a[i][k] > l && a[i][k] < 0) {

l = a[i][k];

}

}

cout << endl;

}

if (l != 0) {

cout << "Наибольшим отрицательным элементом является " << l << endl;

}

else {

cout << "Отрицательных элементов нет. " << endl;

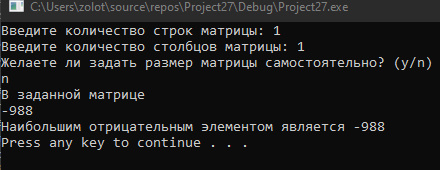
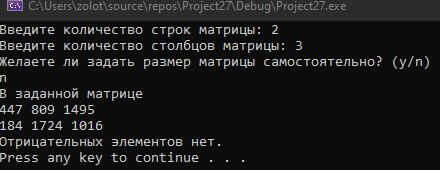
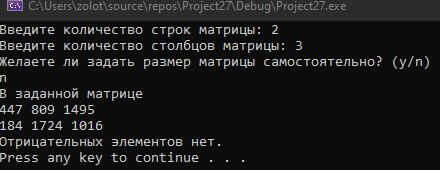
}

system("pause");

return 0;

}

5. Результат выполнения работы программы:

Отладочный пример №1 Отладочный пример №2 Отладочный пример №3

6. Вывод:

В ходе выполнения работы были изучены способы работы с матрицами с помощью языка высокого уровня посредством использования динамического двумерного массива, циклов и условий.

Дважды была проделана ручная отладка блок-схемы, все возникшие ошибки были проанализированы и немедленно устранены.

На контрольных примерах мы убедились, что сейчас наш код работает согласно блок-схеме и успешно проходит все её ветки.

Был оформлен комплект документации на программный код.

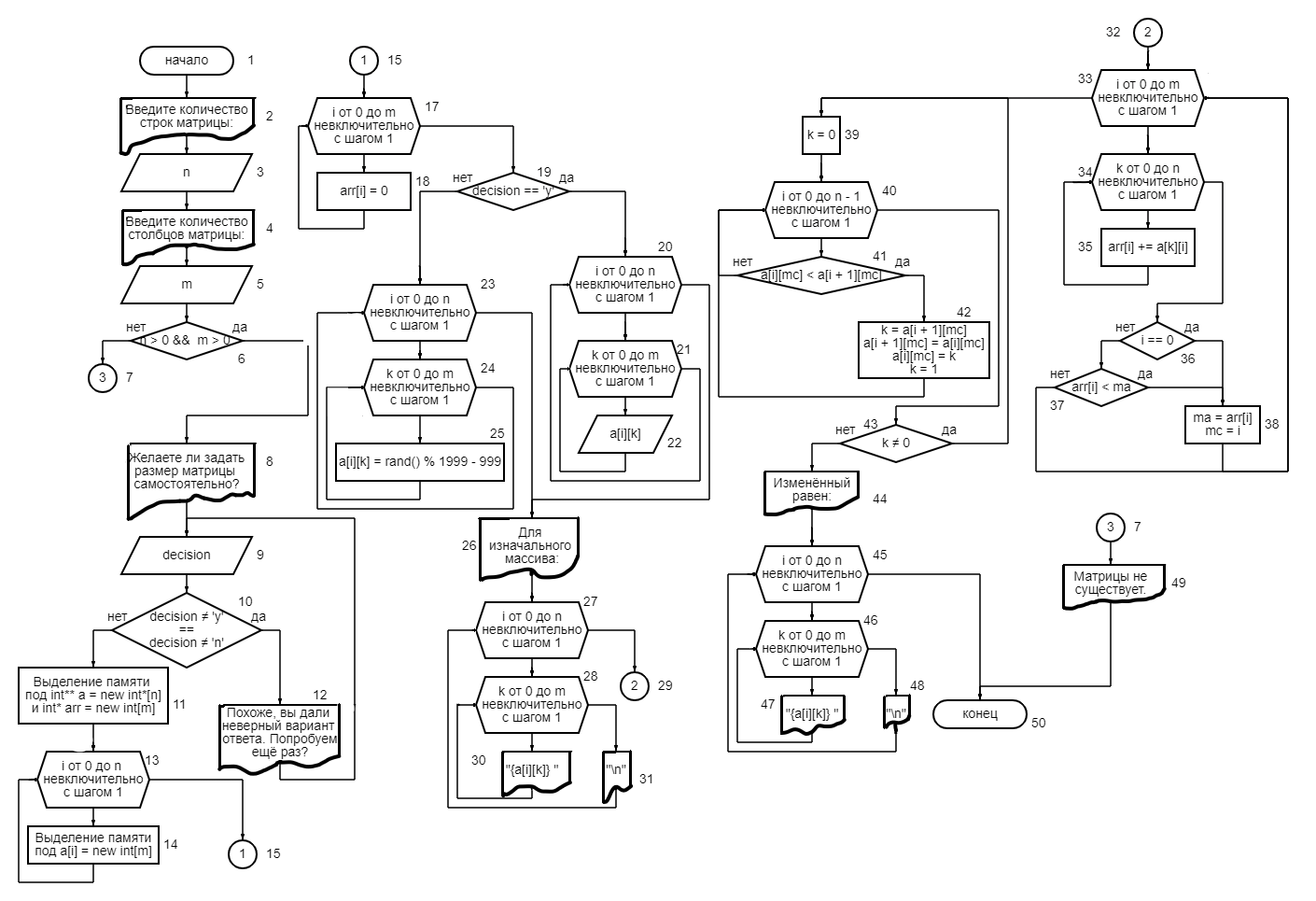
**Задание №7**

Задание: Задана матрица целых чисел. Выполнить сортировку элементов по убыванию в столбце с минимальной суммой элементов.

1. Таблица имён:

|  |  |  |
| --- | --- | --- |
| Исходные данные | | |
| n | целочисленный | Количество строк в матрице |
| m | целочисленный | Количество столбцов в матрице |
| a[n][m] | целочисленный | Исходная матрица |
| decision | символьный | Определитель выбора ввода матрицы |
| Рабочие переменные | | |
| arr[m] | целочисленный | Массив сумм столбцов матрицы |
| a[i][k] | целочисленный | Элемент матрицы i строки j столбца |
| ma | целочисленный | Минимальная сумма в столбце |
| mc | целочисленный | Индекс столбца с минимальной суммой |
| Результат | | |
| a[n][m] | целочисленный | Отсортированная исходная матрица |

2. Блок-схема:



3. Отладочный пример:

Отладочный пример №1:  
1. Начало  
2. Вывод «Введите количество строк матрицы:»  
3. Ввод n = 0  
4. Вывод «Введите количество столбцов матрицы»  
5. Ввод m = 0  
6. 0 > 0 && 0 > 0, нет, переход в блок 7.  
7. Ссылка на блок 49  
49. Вывод «Матрицы не существует. »   
50. Конец

Отладочный пример №2:  
1. Начало  
2. Вывод «Введите количестве строк в матрице »  
3. Ввод n = 3  
4. Вывод «Введите количестве столбцов в матрице »  
5. Ввод m = 3  
6. 3 > 0 && 3 > 0, да, переход в блок 8  
8. Вывод «Желаете ли вы задать размер матрицы соответственно?»  
9. Ввод decision = “Error”  
10. “Error” != ‘y’ == “Error” != ‘n’, да, идём в блок 12  
12. Вывод «Похоже, вы дали неверный вариант ответа. Попробуем ещё раз?»  
9. Ввод 'y'  
10. “y” != ‘y’ == “y” != ‘n’, нет, идём в блок 11  
11. Выделение памяти массиву массивов а размерности 3.  
Выделение памяти массиву arr размерности 3.  
13. Вход в цикл: Переменная i = 0  
14. Выделение памяти элементу массива массивов a[0] размерности 3.  
13. Вход в цикл: Переменная i = 1  
14. Выделение памяти элементу массива массивов a[1] размерности 3.  
13. Вход в цикл: Переменная i = 2  
14. Выделение памяти элементу массива массивов a[2] размерности 3.  
13. Вход в цикл: Переменная i = 3. Граница достигнута, переход в блок 15  
15. Ссылка на блок 17  
17. Вход в цикл: Переменная i = 0.  
18. Переприсваивание переменных arr[0] = 0.  
17. Вход в цикл: Переменная i = 1.  
18. Переприсваивание переменных arr[1] = 0.  
17. Вход в цикл: Переменная i = 2.  
18. Переприсваивание переменных arr[2] = 0.  
17. Вход в цикл. Переменная i = 3. Граница достигнута, переход к блоку 19  
19. 'y' == 'y', да, идём в блок 20  
20. Вход в цикл: Переменная i = 0  
21. Вход в цикл: Переменная k = 0  
22. Ввод a[0][0] = 2  
21. Вход в цикл: Переменная k = 1  
22. Ввод a[0][1] = 7  
21. Вход в цикл: Переменная k = 2. Граница достигнута, переход к блоку 20  
22. Ввод a[0][2] = 3  
20. Вход в цикл: Переменная i = 1  
21. Вход в цикл: Переменная k = 0  
22. Ввод a[1][0] = 5  
21. Вход в цикл: Переменная k = 1  
22. Ввод a[1][1] = 1  
21. Вход в цикл: Переменная k = 2. Граница достигнута, переход к блоку 20  
22. Ввод a[1][2] = 6  
20. Вход в цикл: Переменная i = 2  
21. Вход в цикл: Переменная k = 0  
22. Ввод a[2][0] = 8  
21. Вход в цикл: Переменная k = 1  
22. Ввод a[2][1] = 4  
21. Вход в цикл: Переменная k = 2. Граница достигнута, переход к блоку 20  
22. Ввод a[2][2] = 9  
20. Вход в цикл: Переменная i = 3. Граница достигнута, переход к блоку 26  
26. Вывод «Для изначального массива: »  
27. Вход в цикл: Переменная i = 0  
28. Вход в цикл: Переменная k = 0  
30. Вывод «2 »  
28. Вход в цикл: Переменная k = 1  
30. Вывод «7»  
28. Вход в цикл: Переменная k = 2  
30. Вывод «3 »  
28. Вход в цикл: Переменная k = 3. Граница достигнута, переход к блоку 27  
27. Вход в цикл: Переменная i = 1  
28. Вход в цикл: Переменная k = 0  
30. Вывод «5 »  
28. Вход в цикл: Переменная k = 1  
30. Вывод «1 »  
28. Вход в цикл: Переменная k = 2  
30. Вывод «6 »  
28. Вход в цикл: Переменная k = 3. Граница достигнута, переход к блоку 27  
27. Вход в цикл: Переменная i = 2  
28. Вход в цикл: Переменная k = 0  
30. Вывод «8 »  
28. Вход в цикл: Переменная k = 1  
30. Вывод «4 »  
28. Вход в цикл: Переменная k = 2  
30. Вывод «9 »  
28. Вход в цикл: Переменная k = 3. Граница достигнута, переход к блоку 27  
27. Вход в цикл: Переменная i = 3. Граница достигнута, переход к блоку 29  
29. Ссылка на блок 33  
33. Вход в цикл: Переменная i = 0  
34. Вход в цикл: Переменная k = 0  
35. arr[0] += 2 = 0 + 2 = 2  
34. Вход в цикл: Переменная k = 1  
35. arr[0] += 5 = 2 + 5 = 7  
34. Вход в цикл: Переменная k = 2  
35. arr[0] += 8 = 7 + 8 = 15  
34. Вход в цикл: Переменная k = 3. Граница достигнута, переход в блок 36  
36. 0 == 0, да, переход в блок 38  
38. Переприсваивание переменных ma = 15, mc = 0  
33. Вход в цикл. Переменная i = 1  
34. Вход в цикл: Переменная k = 0  
35. arr[1] += 7 = 0 + 7 = 7  
34. Вход в цикл: Переменная k = 1  
35. arr[1] += 1 = 7 + 1 = 8  
34. Вход в цикл: Переменная k = 2  
35. arr[1] += 4 = 8 + 4 = 12  
34. Вход в цикл: Переменная k = 3. Граница достигнута, переход в блок 36  
36. 1 == 0, нет, переход в блок 37  
37. 12 < 15, да, переход в блок 38  
38. Переприсваивание переменных ma = 12, mc = 1  
33. Вход в цикл. Переменная i = 2  
34. Вход в цикл: Переменная k = 0  
35. arr[2] += 3 = 0 + 3 = 3  
34. Вход в цикл: Переменная k = 1  
35. arr[2] += 6 = 3 + 6 = 9  
34. Вход в цикл: Переменная k = 2  
35. arr[2] += 9 = 9 + 9 = 18  
34. Вход в цикл: Переменная k = 3. Граница достигнута, переход в блок 36  
36. 2 == 0, нет, переход в блок 37  
37. 18 < 12, нет, переход в блок 33  
33. Вход в цикл. Переменная i = 2. Граница достигнута, переход в блок 39  
39. Переприсваивание переменных k = 0  
40. Вход в цикл: Переменная i = 0  
41. 7 < 1, нет, идём в блок 40  
40. Вход в цикл: Переменная i = 1  
41. 1 < 4, да, идём в блок 42  
42. Переприсваивание переменных  
k = 4; a[2][1] = 1; a[1][1] = 4; k = 1  
40. Вход в цикл: Переменная i = 2. Граница достигнута, переход в 43  
43. 1 != 0, нет, идём в блок 39  
39. Переприсваивание переменных k = 0  
40. Вход в цикл: Переменная i = 0  
41. 7 < 4, нет, идём в блок 40  
40. Вход в цикл: Переменная i = 1  
41. 4 < 1, нет, идём в блок 40  
40. Вход в цикл: Переменная i = 2. Граница достигнута, переход в 43  
43. 0 != 0, нет, переход в блок 44  
44. Вывод «Изменённый массив равен: »  
45. Вход в цикл: Переменная i = 0  
46. Вход в цикл: Переменная k = 0  
47. Вывод «2 »  
46. Вход в цикл: Переменная k = 1  
47. Вывод «5 »  
46. Вход в цикл: Переменная k = 2  
47. Вывод «8 »  
46. Вход в цикл: Переменная k = 3. Граница достигнута, переход в блок 48  
48. Перенос текста на новую строку  
45. Вход в цикл: Переменная i = 1  
46. Вход в цикл: Переменная k = 0  
47. Вывод «7 »  
46. Вход в цикл: Переменная k = 1  
47. Вывод «4 »  
46. Вход в цикл: Переменная k = 2  
47. Вывод «1 »  
46. Вход в цикл: Переменная k = 3. Граница достигнута, переход в блок 48  
48. Перенос текста на новую строку  
45. Вход в цикл: Переменная i = 2  
46. Вход в цикл: Переменная k = 0  
47. Вывод «3 »  
46. Вход в цикл: Переменная k = 1  
47. Вывод «6 »  
46. Вход в цикл: Переменная k = 2  
47. Вывод «9 »  
46. Вход в цикл: Переменная k = 3. Граница достигнута, переход в блок 48  
48. Перенос текста на новую строку  
45. Вход в цикл: Переменная i = 3. Граница достигнута, переход в блок 50  
50. Конец  
  
Отладочный пример №3

1. Начало  
2. Вывод «Введите количество строк матрицы»  
3. Ввод n = 2  
4. Вывод «Введите количество столбцов в матрице»  
5. Ввод m = 2  
6. 2 > 0 && 2 > 0, да, переход к блоку 8   
8. Вывод «Желаете ли задать размер матрицы самостоятельно7»  
9. Ввод decision = 'n'  
10. decision != y && decision != 'n', нет, переход в блок 11  
11. Выделение памяти под массив массивов a размерности 2  
Выделение памяти под массив сумм arr размерности 2  
13. Вход в цикл: Переменная i = 0  
14. Выделение памяти элементу массива массивов a[0] размерности 2  
13. Вход в цикл: Переменная i = 1  
14. Выделение памяти элементу массива массивов a[1] размерности 2  
13. Вход в цикл: Переменная i = 2. Граница достигнута, переход в блок 15  
15. Ссылка на блок 17  
17. Вход в цикл: Переменная i = 0  
18. Переприсваивание переменных arr[0] = 0  
17. Вход в цикл: Переменная i = 1  
18. Переприсваивание переменных arr[1] = 0  
17. Вход в цикл: Переменная i = 2. Граница достигнута, переход в блок 19.  
19. ‘n’ == ‘y’, нет, переход к блоку 23.  
23. Вход в цикл: Переменная i = 0  
24. Вход в цикл: Переменная k = 0  
25. Переприсваивание переменных a[0][0] = 283  
24. Вход в цикл: Переменная k = 1  
25. Переприсваивание переменных a[0][1] = 200  
24. Вход в цикл: Переменная k = 2. Граница достигнута, переход в блок 23  
23. Вход в цикл: Переменная i = 0  
24. Вход в цикл: Переменная k = 1  
25. Переприсваивание переменных a[1][0] = 115  
24. Вход в цикл: Переменная k = 1  
25. Переприсваивание переменных a[1][1] = 639  
24. Вход в цикл: Переменная k = 2. Граница достигнута, переход в блок 23  
23. Вход в цикл: Переменная i = 2. Граница достигнута, переход в блок 26  
26. Вывод «Для изначального массива »  
23. Вход в цикл: Переменная i = 0  
24. Вход в цикл: Переменная k = 0  
25. Вывод «283 »  
24. Вход в цикл: Переменная k = 1  
25. Вывод «200 »  
24. Вход в цикл: Переменная k = 2. Граница достигнута, переход в блок 31  
31. Перевод текста на новую строку  
23. Вход в цикл: Переменная i = 0  
24. Вход в цикл: Переменная k = 1  
25. Вывод «115»  
24. Вход в цикл: Переменная k = 1  
25. Вывод «639»  
24. Вход в цикл: Переменная k = 2. Граница достигнута, переход в блок 31  
31. Перевод текста на новую строку  
23. Вход в цикл: Переменная i = 2. Граница достигнута, переход в блок 29  
29. Ссылка на блок 33  
33. Вход в цикл: Переменная i = 0  
34. Вход в цикл: Переменная k = 0  
35. Переприсваивание переменных arr[0] += a[0][0] = 0 + 283 = 283  
34. Вход в цикл: Переменная k = 1  
35. Переприсваивание переменных arr[0] += a[0][1] = 283 + 115 = 398  
34. Вход в цикл: Переменная k = 2, граница достигнута, переход в блок 36  
36. 0 == 0, да, переход к блоку 38  
38 Переприсваивание переменных ma = 398, mc = 0  
33. Вход в цикл: Переменная i = 1  
34. Вход в цикл: Переменная k = 0  
35. Переприсваивание переменных arr[1] += a[1][0] = 0 + 200 = 200  
34. Вход в цикл: Переменная k = 1  
35. Переприсваивание переменных arr[1] += a[1][1] = 200 + 639 = 839  
34. Вход в цикл: Переменная k = 2, граница достигнута, переход в блок 36  
36. 1 == 0, нет, переход у блоку 37  
37. 839 < 398, нет, переход к блоку 33.  
33. Вход в цикл: Переменная i = 2. Граница достигнута, переход к блоку 39  
39. Переприсваивание переменных k = 0  
40. Вход в цикл: Переменная i = 0  
41. 283 < 115, нет, переход к блоку 40  
40. Вход в цикл: Переменная i = 1. Граница достигнута, переход к блоку 43  
43ю 0 != 0, нет, переход к блоку 44  
44. Вывод «Изменённый равен: »  
45. Вход в цикл: Переменная i = 0  
46. Вход в цикл: Переменная k = 0  
47. Вывод «283 »  
46. Вход в цикл: Переменная k = 1  
47. Вывод «200 »  
46. Вход в цикл: Переменная k = 2. Граница достигнута, переход в блок 48  
48. Перевод текста на новую строку  
45. Вход в цикл: Переменная i = 0  
46. Вход в цикл: Переменная k = 1  
47. Вывод «115»  
46. Вход в цикл: Переменная k = 1  
47. Вывод «639»  
46. Вход в цикл: Переменная k = 2. Граница достигнута, переход в блок 48  
45. Вход в цикл: Переменная i = 2. Граница достигнута, переход в блок 50  
50. Конец

4. Код программы:

#include <iostream>

#include <ctime>

using namespace std;

int main() {

srand(time(NULL)); // Для того, чтобы rand() генерировала постоянно новые числа

setlocale(LC\_ALL, "rus");

int n, m, ma, mc, i, k; char decision;

cout << "Введите количество строк матрицы: ";

cin >> n;

cout << "Введите количество столбцов матрицы: ";

cin >> m;

cout << "Желаете ли задать размер матрицы самостоятельно? (y/n)" << endl;

cin >> decision;

while (decision != 'y' && decision != 'n') {

cout << "Похоже, вы дали неверный вариант ответа. Попробуем ещё раз?\n";

cout << "Желаете ли задать элементы матрицы самостоятельно? (y/n)" << endl;

cin >> decision;

}

if (n <= 0 || m <= 0) {

cout << "Матрицы не существует." << endl;

system("pause");

return 0;

}

int\*\* a = new int\* [n], \* arr = new int[m];

//Присваиваем каждому элементу массива размерность m

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

a[i] = new int[m];

}

//Присваиваем каждому элементу массива суммы значение 0

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

arr[i] = 0;

}

// Ввод массива пользователем

if (decision == 'y') {

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

for (k = 0; k < m; k++) {

cin >> a[i][k];

}

}

}

// Ввод массива компьютером

else {

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

for (k = 0; k < m; k++) {

a[i][k] = rand() % 1999 - 999;

}

}

}

cout << "Для изначального массива: " << endl;

// Вывод введенного массива

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

for (k = 0; k < m; k++) {

cout << a[i][k] << " ";

}

cout << endl;

}

//Определение столбца с минимальной суммой

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

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

arr[i] += a[k][i];

}

if (i == 0) {

ma = arr[i];

mc = i;

}

else if (arr[i] < ma) {

ma = arr[i];

mc = i;

}

}

// Сортировка столбца пузырьком

do {

k = 0;

for (i = 0; i < n - 1; i++) {

if (a[i][mc] < a[i + 1][mc]) {

k = a[i + 1][mc];

a[i + 1][mc] = a[i][mc];

a[i][mc] = k;

k = 1;

}

}

} while (k != 0);

cout << "Изменённый равен: " << endl;

//Вывод изменённого массива

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

for (k = 0; k < m; k++) {

cout << a[i][k] << " ";

}

cout << endl;

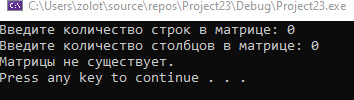
}

system("pause");

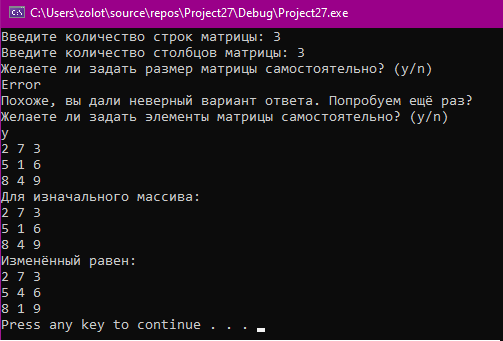
return 0;

}

5. Результат выполнения работы программы:

 kjiibibib

ede Отладочный пример №1



Отладочный пример №2

A screenshot of a computer

Description automatically generated

Отладочный пример №3

**6.** Вывод:

В ходе выполнения работы были изучены способы заполнения, обработки двумерных динамических массивов, их выборочную сортировку, а также вывода. Были устранены все исключения, возникшие по ходу работы.

Было проделано исследование способов реализации такой программы с помощью различных учебных материалов

На контрольных примерах мы убедились, что программа работает безошибочно, независимо от введённых данных.

Был оформлен комплект документации на программный код.

**Задание №8**

Задание: задана матрица целых чисел. Определить количество максимальных элементов в ней.

1. Таблица имён:

|  |  |  |
| --- | --- | --- |
| Исходные данные | | |
| a | целочисленный | Количество столбцов в матрице |
| b | целочисленный | Количество строк в матрице |
| s[a][b] | целочисленный | Двумерный массив, содержащий матрицу |
| Рабочие переменные | | |
| s[i] | Указатель целочисленного типа | Последовательное выделение памяти матрице для организовывания массивов в массиве |
| maximum | целочисленный | Максимальный элемент в матрице |
| Результат | | |
| counter | целочисленный | Количество максимальных элементов в матрице |

2. Блок-схема:

Diagram

Description automatically generated

3. Отладочный пример:

Вариант I

1. Начало  
2. Переприсваивание переменных  
3. Вывод: Введите количество строк в матрице  
4. Ввод а = 3  
5. Вывод: Введите количество столбцов в матрице  
6. Ввод b = 3  
10. 3 < 1 || 3 < 1, нет, переход к блоку 7  
7. Выделение памяти под массив массивов s размерности 3  
8. Вход в цикл: Переменная i = 0  
9. Выделение памяти под элемент массив массивов s[0] размерности 3  
8. Вход в цикл: Переменная i = 1  
9. Выделение памяти под элемент массив массивов s[1] размерности 3  
8. Вход в цикл: Переменная i = 2  
9. Выделение памяти под элемент массив массивов s[2] размерности 3  
8. Вход в цикл: Переменная i = 3. Граница достигнута, переход к блоку 10  
12. Вывод «Желаете ли задать элементы матрицы самостоятельно? (y/n) »  
13. Ввод decision = 'k'  
14. Ссылка на блок 16  
16. 'k' != 'y' && 'k' != 'n', да, переход к блоку 17  
17. Вывод «Похоже, вы дали неправильный вариант ответа. Попробуем ещё раз? »  
18. Вывод «Желаете ли задать элементы матрицы самостоятельно? »  
19. Ввод decision = 'y'  
16. 'y' != 'y' && 'y' != 'n', нет, переход к блоку 20  
20. 'y' == 'y', да, переход к блоку 21  
21. Вывод «Введите элементы матрицы: »  
22. Вход в цикл: Переменная i = 0  
23. Вход в цикл: Переменная k = 0  
24. Ввод a[0][0] = 0  
23. Вход в цикл: Переменная k = 1  
24. Ввод a[0][1] = 9   
23. Вход в цикл: Переменная k = 2  
24. Ввод a[0][2] = 0   
23. Вход в цикл: Переменная k = 3. Граница достигнута, переход к блоку 22.  
22. Вход в цикл: Переменная i = 1  
23. Вход в цикл: Переменная k = 0  
24. Ввод a[1][0] = 9   
23. Вход в цикл: Переменная k = 1  
24. Ввод a[1][1] = 0   
23. Вход в цикл: Переменная k = 2  
24. Ввод a[1][2] = 9   
23. Вход в цикл: Переменная k = 3. Граница достигнута, переход к блоку 22.  
22. Вход в цикл: Переменная i = 2  
23. Вход в цикл: Переменная k = 0  
24. Ввод a[2][0] = 0  
23. Вход в цикл: Переменная k = 1  
24. Ввод a[2][1] = 9   
23. Вход в цикл: Переменная k = 2  
24. Ввод a[2][2] = 0   
23. Вход в цикл: Переменная k = 3. Граница достигнута, переход к блоку 22.  
22. Вход в цикл: Переменная i = 3. Граница достигнута, переход к блоку 26.  
26. Переприсваивание переменных maximum = 0  
29. Ссылка на блок 30.  
30. Вход в цикл: Переменная i = 0  
31. Вход в цикл: Переменная k = 0   
32. 0 == 0 && 0 == 0, да, переход к блоку 31  
31. Вход в цикл: Переменная k = 1  
32. 0 == 0 && 1 == 0, нет, переход к блоку 33  
33. 9 == 0, нет, переход к блоку 35  
35. 9 > 0, да, переход к блоку 36  
36. Переприсваивание переменных maximum = 9, counter = 1  
31. Вход в цикл: Переменная k = 2  
32. 0 == 0 && 2 == 0, нет, переход к блоку 33  
33. 0 == 9, нет, переход к блоку 35  
35. 0 > 9, нет, переход к блоку 31  
31. Вход в цикл: Переменная k = 3. Граница достигнута, переход к блоку 30  
30. Вход в цикл: Переменная i = 1  
31. Вход в цикл: Переменная k = 0   
32. 1 == 0 && 0 == 0, нет, переход к блоку 33  
33. 9 == 9, да, переход к блоку 34  
34. Переприсваивание переменных counter++ -> 1 + 1 = 2  
35. 9 > 9, нет, переход к блоку 31  
31. Вход в цикл: Переменная k = 1  
32. 1 == 0 && 1 == 0, нет, переход к блоку 33  
33. 0 == 9, нет, переход к блоку 35  
35. 9 > 9, нет, переход к блоку 31  
31. Вход в цикл: Переменная k = 2  
32. 1 == 0 && 2 == 0, нет, переход к блоку 33  
33. 9 == 9, да, переход к блоку 34  
34. Переприсваивание переменных counter++ -> 2 + 1 = 3  
35. 9 > 9, нет, переход к блоку 31  
31. Вход в цикл: Переменная k = 3. Граница достигнута, переход к блоку 30  
30. Вход в цикл: Переменная i = 2  
31. Вход в цикл: Переменная k = 0   
32. 2 == 0 && 0 == 0, нет, переход к блоку 33  
33. 0 == 9, нет переход к блоку 35  
35. 0 > 9, нет, переход к блоку 31  
31. Вход в цикл: Переменная k = 1  
32. 2 == 0 && 1 == 0, нет, переход к блоку 34  
34. Переприсваивание переменных counter++ -> 3 + 1 = 4  
35. 9 > 9, нет, переход к блоку 31  
31. Вход в цикл: Переменная k = 2  
32. 2 == 0 && 2 == 0, нет, переход к блоку 33  
33. 0 == 9, нет, переход к блоку 35  
35. 0 > 9, нет, переход к блоку 31  
31. Вход в цикл: Переменная k = 3. Граница достигнута, переход в блок 30.  
30. Вход в цикл: Переменная i = 3. Граница достигнута, переход в блок 37.  
37. Вывод «Максимальный элемент в матрице – 9,   
его количество в матрице - 4»  
38. Конец

Вариант II

1. Начало  
2. Переприсваивание переменных counter = 1  
3. Вывод «Введите количество строк в матрице: »  
4. Ввод a = 0  
5. Вывод «Введите количество столбцов в матрице»  
6. Ввод b = -5  
10. 0 < 1 || -5 < 1, да, переход к блоку 11  
11. Вывод «Матрица нулевой или отрицательной   
размерности не имеет элементов»  
15. Ссылка на блок 38  
38. Конец

Вариант III

1. Начало  
2. Переприсваивание переменных counter = 1  
3. Вывод «Введите количество строк в матрице: »  
4. Ввод a = 2  
5. Вывод «Введите количество столбцов в матрице: »  
6. Ввод b = 2  
10. 2 < 1 || 2 < 1, нет, переход к блоку 7  
7. Выделение памяти под массив массивов s размерности 2  
8. Вход в цикл: Переменная i = 0  
9. Выделение памяти под элемент массива массивов s[0] размерности 2  
8. Вход в цикл: Переменная i = 1  
9. Выделение памяти под элемент массива массивов s[1] размерности 2  
8. Вход в цикл: Переменная i = 2. Граница достигнута, переход к блоку 12  
12. Вывод «Желаете ли задать элементы матрицы самостоятельно? (y/n)»  
13. Ввод decision = 'n'  
14. Ссылка на блок 16  
16. ‘n’ != 'y' && ‘n’ != 'n', нет, переход к блоку 20  
20. 'n' == 'y', нет, переход к блоку 25  
25. Вход в цикл: Переменная i = 0  
26. Вход в цикл: Переменная k = 0  
27. Переприсваивание переменных a[0][0] = 835  
26. Вход в цикл: Переменная k = 1  
27. Переприсваивание переменных a[0][1] = 861   
26. Вход в цикл: Переменная k = 2. Граница достигнута, переход к блоку 25  
25. Вход в цикл: Переменная i = 1  
26. Вход в цикл: Переменная k = 0  
27. Переприсваивание переменных a[1][0] = 35  
26. Вход в цикл: Переменная k = 1  
27. Переприсваивание переменных a[1][1] = -505  
26. Вход в цикл: Переменная k = 2. Граница достигнута, переход к блоку 25  
25. Вход в цикл: Переменная i = 2. Граница достигнута, переход к блоку 28  
28. Переприсваивание переменных maximum = 835  
29. Ссылка на блок 30  
30. Вход в цикл: Переменная i = 0  
31. Вход в цикл: Переменная k = 0  
32. 0 == 0 && 0 == 0, да, переход к блоку 31  
31. Вход в цикл: Переменная k = 1  
32. 0 == 0 && 1 == 0, нет, переход к блоку 33  
33. 861 == 835, нет, переход к блоку 35  
35. 861 > 835, да, переход к блоку 36  
36. Переприсваивание переменных maximum = 861, counter = 1  
31. Вход в цикл: Переменная k = 2. Граница достигнута, переход к блоку 30  
30. Вход в цикл: Переменная i = 1  
31. Вход в цикл: Переменная k = 0  
32. 1 == 0 && 0 == 0, нет, переход к блоку 33  
33. 35 == 861, нет, переход к блоку 35  
35. 35 > 861, нет, переход к блоку 31.  
31. Вход в цикл: Переменная k = 1  
32. 1 == 0 && 1 == 0, нет, переход к блоку 33  
33. -505 == 861, нет, переход к блоку 35  
35. -505 > 861, нет, переход к блоку 31.  
31. Вход в цикл: Переменная k = 2. Граница достигнута, переход к блоку 30.  
30. Вход в цикл: Переменная i = 2. Граница достигнута, переход к блоку 37.  
37. Вывод «Максимальный элемент в матрице – 861, его количество в матрице - 1»  
38. Конец

4. Код программы:

#include <iostream>

#include <ctime>

using namespace std;

int main() {

srand(time(0));

setlocale(LC\_ALL, "rus");

int a, b, \*\* s, maximum, counter = 1;

char decision;

cout << "Введите количество строк в матрице: ";

cin >> a;

cout << "Введите количество столбцов в матрице: ";

cin >> b;

s = new int\* [a];

//Выделение памяти под массивыв массиве массивов

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

s[i] = new int[b];

// Проверка на существование

if (a < 1 || b < 1) {

cout << "Матрица нулевой или отрицательной размерности не имеет элементов" << endl;

system("pause");

return 0;

}

// Выбор способа заполнения

cout << "Желаете ли задать элементы матрицы самостоятельно? (y/n)" << endl;

cin >> decision;

while (decision != 'y' && decision != 'n') {

cout << "Похоже, вы дали неверный вариант ответа. Попробуем ещё раз?\n";

cout << "Желаете ли задать элементы матрицы самостоятельно? (y/n)" << endl;

cin >> decision;

}

// Заполнение пользователем

if (decision == 'y') {

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

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

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

cin >> s[i][k];

}

}

}

// Заполнение компьютером

else {

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

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

s[i][k] = rand() % 1999 - 999;

}

}

}

//Определение максимума и его подсчёт

maximum = s[0][0];

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

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

if (i == 0 && k == 0)

continue;

if (s[i][k] == maximum)

counter++;

if (s[i][k] > maximum) {

maximum = s[i][k];

counter = 1;

}

}

}

printf("Максимальный элемент в матрице - %d, его количество в матрице - %d\n", maximum, counter);

system("pause");

return 0;

}

5. Результат выполнения работы программы:

Text

Description automatically generated

Отладочный пример №1

Text

Description automatically generated

Отладочный пример №2

Text

Description automatically generated

Отладочный пример №3

6. Вывод:

В ходе выполнения работы были изучены способы анализа матрицы, заданной в программе в виде двумерного массива, а также перебор её компонентов с помощью использования циклов вида «for».

Было проделано необходимое количество отладок для выявления исключений и их последующего устранения на программном уровне.

На контрольных примерах мы убедились, что код работает безошибочно.

Был оформлен комплект документации на программный код.

**Задание №9**

Задание: задан текст. Между словами пробел, за последним словом точка. Определить порядковый номер слова заданного текста.

1. Таблица имён:

|  |  |  |
| --- | --- | --- |
| Исходные данные | | |
| main\_text | Символьный динамический массив | Исходный текст, заданный пользователем |
| main\_word | Символьный динамический массив | Слово, заданное пользователем, которое будет искаться в заданном тексте. |
| Рабочие переменные | | |
| len | Целочисленный | Переменная для увеличения размерности символьных динамических массивов со считыванием каждого символа с консоли. |
| c | Символьный | Переменная для последовательного считывания символов с консоли |
| something | Символьный динамический массив | Переменная для обработки исходных данных типа символьных массивов с замещением заглавных букв на строчные |
| something[i] | Символьный | Переменная для установления, является ли буква заглавной или строчной |
| s | Символьный динамический массив | Постепенно увеличивающийся массив символов, который увеличивается соответственно с вводимыми пользователем символами |
| fact | Целочисленный | Переменная-флаг, обозначающая обнаружение слова в полном соответствии в заданном тексте |
| sequence | Целочисленный | Переменная для сравнения слов на полное соответствие друг другу, анализирует только символы текста, но не слова. |
| position | Целочисленный | Местоположение вхождения первого символа найденного слова в тексте |
| i | Целочисленный | Переменная-счётчик в циклах |
| k | Целочисленный | Переменная-счётчик в циклах |
| Результат | | |
| counter | Целочисленный | Переменная-счётчик, отвечает за подсчёт пробелов перед найденным словом, таким образом находя порядковый номер слова. |

2. Блок-схема:

Diagram

Description automatically generated

3. Отладочный пример:

Вариант I

1. Начало  
38. Переприсваивание переменных counter = 0  
2. Вывод «Введите текст: »  
3. Переходим к выполнению подпрограммы read\_string():  
20. Работа с переменными:  
len = 1, выделяем память под s (1 байт), считываем введённым пользователем символ с консоли в переменную с = 'М'  
21. 'М' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (1 байт), s[0] = 'M', считываем в переменную с следующий символ с консоли 'А', len++ = 1+1=2  
21. 'A' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (2 байта), s[1] = 'A', считываем в переменную с следующий символ с консоли 'ш', len++ = 2+1=3  
21. 'ш' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (3 байта), s[2] = 'ш', считываем в переменную с следующий символ с консоли 'А', len++ = 3+1=4  
21. 'A' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (4 байта), s[3] = 'A', считываем в переменную с следующий символ с консоли 'ш', len++ = 4+1=5  
21. ' ' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (5 байтов), s[4] = ' ', считываем в переменную с следующий символ с консоли 'в', len++ = 5+1=6  
21. 'в ' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (6 байтов), s[5] = 'в', считываем в переменную с следующий символ с консоли 'Ы', len++ = 6+1=7  
21. 'Ы ' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (7 байтов), s[6] = 'Ы', считываем в переменную с следующий символ с консоли 'ш', len++ = 7+1=8  
21. 'ш ' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (8 байтов), s[7] = 'ш', считываем в переменную с следующий символ с консоли 'Л', len++ = 8+1=9  
21. 'Л ' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (9 байтов), s[8] = 'Л', считываем в переменную с следующий символ с консоли 'а', len++ = 9+1=10  
21. 'а ' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (10 байтов), s[9] = 'а', считываем в переменную с следующий символ с консоли ' ', len++ = 10+1=11  
21. ' ' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (11 байтов), s[10] = ' ', считываем в переменную с следующий символ с консоли 'п', len++ = 11+1=12  
21. 'п' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (12 байтов), s[11] = 'п', считываем в переменную с следующий символ с консоли 'о', len++ = 12+1=13  
21. 'о ' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (13 байтов), s[12] = 'о', считываем в переменную с следующий символ с консоли 'г', len++ = 13+1=14  
21. 'г' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (14 байтов), s[13] = 'г', считываем в переменную с следующий символ с консоли 'у', len++ = 14+1=15  
21. 'у' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (15 байтов), s[14] = 'у', считываем в переменную с следующий символ с консоли 'л', len++ = 15+1=16  
21. 'л' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (16 байтов), s[15] = 'л', считываем в переменную с следующий символ с консоли 'я', len++ = 16+1=17  
21. 'я' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (17 байтов), s[16] = 'я', считываем в переменную с следующий символ с консоли 'т', len++ = 17+1=18  
21. 'т' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (18 байтов), s[17] = 'т', считываем в переменную с следующий символ с консоли 'ь', len++ = 18+1=19  
21. 'ь' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (19 байтов), s[18] = 'ь', считываем в переменную с следующий символ с консоли '.', len++ = 19+1=20  
21. '.' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (20 байтов), s[19] = '.', считываем в переменную с следующий символ с консоли '\n',   
len++ = 20 + 1 = 21  
21. '\n' != '\n', нет, переход к блоку 23  
23. Переприсваивание переменных s[len - 1] = '\0'  
24. Возврат s из подпрограммы  
Переприсваивание переменных main\_text = “МАшА вЫшЛа погулять.”  
4. Вывод «Введите ваше слово: »  
5. Переходим к выполнению подпрограммы read\_string():  
20. Работа с переменными:  
len = 1, выделяем память под s (1 байт), считываем введённым пользователем символ с консоли в переменную с = 'в'  
21. ‘в’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (1 байт), s[0] = ‘в’, считываем в переменную с следующий символ с консоли ‘ы’, len++ = 1+1=2  
21. ‘ы’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (2 байта), s[1] = ‘ы’, считываем в переменную с следующий символ с консоли ‘ш’, len++ = 2+1=3  
21. ‘ш’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (3 байта), s[2] = ‘ш’, считываем в переменную с следующий символ с консоли ‘л’, len++ = 3+1=4  
21. ‘л’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (4 байта), s[3] = ‘л’, считываем в переменную с следующий символ с консоли ‘а’, len++ = 4+1=5  
21. ‘а’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (4 байта), s[4] = ‘а’, считываем в переменную с следующий символ с консоли ‘\n’, len++ = 5+1=6  
21. ‘\n’ != ‘\n’, нет, переход к блоку 23  
23. Переприсваивание переменных s[5] = ‘\0’  
24. Возврат s из подпрограммы  
Переприсваивание переменных main\_word = “вышла”  
6. Переходим к выполнению подпрограммы find(main\_text, main\_word)  
25. Переход к выполнению подпрограммы format(text)  
14. Вход в цикл: Переменная k = 0  
15. ‘M’ >= ‘A’ && ‘M’ <= ‘Z’, нет, переход к блоку 17  
17. ‘M’ >= ‘A’ && ‘M’ <= ‘Я’, да, переход к блоку 18  
18. Переприсваивание переменных something[0] = ‘M’ – ‘A’ + ‘a’ = ‘м’  
14. Вход в цикл: Переменная k = 1  
15. ‘А’ >= ‘A’ && ‘А’ <= ‘Z’, нет, переход к блоку 17  
17. ‘А’ >= ‘A’ && ‘А’ <= ‘Я’, да, переход к блоку 18  
18. Переприсваивание переменных something[1] = ‘А’ – ‘A’ + ‘a’ = ‘а’  
14. Вход в цикл: Переменная k = 2  
15. ‘ш’ >= ‘A’ && ‘ш’ <= ‘Z’, нет, переход к блоку 17  
17. ‘ш’ >= ‘A’ && ‘ш’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 3  
15. ‘А’ >= ‘A’ && ‘А’ <= ‘Z’, нет, переход к блоку 17  
17. ‘А’ >= ‘A’ && ‘А’ <= ‘Я’, да, переход к блоку 18  
18. Переприсваивание переменных something[3] = ‘А’ – ‘A’ + ‘a’ = ‘а’  
14. Вход в цикл: Переменная k = 4  
15. ‘ ’ >= ‘A’ && ‘ ’ <= ‘Z’, нет, переход к блоку 17  
17. ‘ ’ >= ‘A’ && ‘ ’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 5  
15. ‘в’ >= ‘A’ && ‘в’ <= ‘Z’, нет, переход к блоку 17  
17. ‘в’ >= ‘A’ && ‘в’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 6  
15. ‘Ы’ >= ‘A’ && ‘Ы’ <= ‘Z’, нет, переход к блоку 17  
17. ‘Ы’ >= ‘A’ && ‘Ы’ <= ‘Я’, да, переход к блоку 18  
18. Переприсваивание переменных something[6] = ‘Ы’ – ‘A’ + ‘a’ = ‘ы’  
14. Вход в цикл: Переменная k = 7  
15. ‘ш’ >= ‘A’ && ‘ш’ <= ‘Z’, нет, переход к блоку 17  
17. ‘ш’ >= ‘A’ && ‘ш’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 8  
15. ‘Л’ >= ‘A’ && ‘Л’ <= ‘Z’, нет, переход к блоку 17  
17. ‘Л’ >= ‘A’ && ‘Л’ <= ‘Я’, да, переход к блоку 18  
18. Переприсваивание переменных something[8] = ‘Л’ – ‘A’ + ‘a’ = ‘л’  
14. Вход в цикл: Переменная k = 9  
15. ‘а’ >= ‘A’ && ‘а’ <= ‘Z’, нет, переход к блоку 17  
17. ‘а’ >= ‘A’ && ‘а’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 10  
15. ‘ ’ >= ‘A’ && ‘ ’ <= ‘Z’, нет, переход к блоку 17  
17. ‘ ’ >= ‘A’ && ‘ ’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 11  
15. ‘п’ >= ‘A’ && ‘п’ <= ‘Z’, нет, переход к блоку 17  
17. ‘п’ >= ‘A’ && ‘п’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 12  
15. ‘о’ >= ‘A’ && ‘о’ <= ‘Z’, нет, переход к блоку 17  
17. ‘о’ >= ‘A’ && ‘о’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 13  
15. ‘г’ >= ‘A’ && ‘г’ <= ‘Z’, нет, переход к блоку 17  
17. ‘г’ >= ‘A’ && ‘г’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 14  
15. ‘у’ >= ‘A’ && ‘у’ <= ‘Z’, нет, переход к блоку 17  
17. ‘у’ >= ‘A’ && ‘у’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 15  
15. ‘л’ >= ‘A’ && ‘л’ <= ‘Z’, нет, переход к блоку 17  
17. ‘л’ >= ‘A’ && ‘л’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 16  
15. ‘я’ >= ‘A’ && ‘я’ <= ‘Z’, нет, переход к блоку 17  
17. ‘я’ >= ‘A’ && ‘я’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 17  
15. ‘т’ >= ‘A’ && ‘т’ <= ‘Z’, нет, переход к блоку 17  
17. ‘т’ >= ‘A’ && ‘т’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 18  
15. ‘ь’ >= ‘A’ && ‘ь’ <= ‘Z’, нет, переход к блоку 17  
17. ‘ь’ >= ‘A’ && ‘ь’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 19  
15. ‘.’ >= ‘A’ && ‘.’ <= ‘Z’, нет, переход к блоку 17  
17. ‘.’ >= ‘A’ && ‘.’ <= ‘Я’, нет, переход к блоку 14  
15. Вход в цикл: Переменная k = 20. Граница достигнута, переход к блоку 19  
19. Возврат something из подпрограммы  
Переприсваивание переменных text = “маша вышла погулять.”  
Переходим к выполнению подпрограммы format(word):  
14. Вход в цикл: Переменная k = 0  
15. ‘в’ >= ‘A’ && ‘в’ <= ‘Z’, нет, переход к блоку 17  
17. ‘в’ >= ‘A’ && ‘в’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 1  
15. ‘ы’ >= ‘A’ && ‘ы’ <= ‘Z’, нет, переход к блоку 17  
17. ‘ы’ >= ‘A’ && ‘ы’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 2  
15. ‘ш’ >= ‘A’ && ‘ш’ <= ‘Z’, нет, переход к блоку 17  
17. ‘ш’ >= ‘A’ && ‘ш’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 3  
15. ‘л’ >= ‘A’ && ‘л’ <= ‘Z’, нет, переход к блоку 17  
17. ‘л’ >= ‘A’ && ‘л’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 4  
15. ‘а’ >= ‘A’ && ‘а’ <= ‘Z’, нет, переход к блоку 17  
17. ‘а’ >= ‘A’ && ‘а’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 5. Граница достигнута, переход к блоку 19  
19. Возврат something из подпрограммы  
Переприсваивание переменных word = “вышла”  
26. Вход в цикл: Переменная i = 0  
27. Переприсваивание переменных fact = 1, sequence = 0  
28. Вход в цикл: Переменная k = 0  
29. ‘в’ != ‘м’, да, переход к блоку 30  
30. Переприсваивание переменных fact = 0  
32. 0 == 1, нет, переход к блоку 26  
26. Вход в цикл: Переменная i = 1  
27. Переприсваивание переменных fact = 1, sequence = 1  
28. Вход в цикл: Переменная k = 0  
29. ‘в’ != ‘а’, да, переход к блоку 30  
30. Переприсваивание переменных fact = 0  
32. 0 == 1, нет, переход к блоку 26  
26. Вход в цикл: Переменная i = 2  
27. Переприсваивание переменных fact = 1, sequence = 2  
28. Вход в цикл: Переменная k = 0  
29. ‘в’ != ‘ш’, да, переход к блоку 30  
30. Переприсваивание переменных fact = 0  
32. 0 == 1, нет, переход к блоку 26  
26. Вход в цикл: Переменная i = 3  
27. Переприсваивание переменных fact = 1, sequence = 3  
28. Вход в цикл: Переменная k = 0  
29. ‘в’ != ‘а’, да, переход к блоку 30  
30. Переприсваивание переменных fact = 0  
32. 0 == 1, нет, переход к блоку 26  
26. Вход в цикл: Переменная i = 4  
27. Переприсваивание переменных fact = 1, sequence = 4  
28. Вход в цикл: Переменная k = 0  
29. ‘в’ != ‘ ’, да, переход к блоку 30  
30. Переприсваивание переменных fact = 0  
32. 0 == 1, нет, переход к блоку 26  
26. Вход в цикл: Переменная i = 5  
27. Переприсваивание переменных fact = 1, sequence = 5  
28. Вход в цикл: Переменная k = 0  
29. ‘в’ != ‘в’, нет, переход к блоку 31  
31. Переприсваивание переменных sequence++ = 5 + 1 = 6  
32. Вход в цикл: Переменная k = 1  
29. ‘ы’ != ‘ы’, нет, переход к блоку 31  
31. Переприсваивание переменных sequence++ = 6 + 1 = 7  
32. Вход в цикл: Переменная k = 2  
29. ‘ш’ != ‘ш’, нет, переход к блоку 31  
31. Переприсваивание переменных sequence++ = 7 + 1 = 8  
32. Вход в цикл: Переменная k = 3  
29. ‘л’ != ‘л’, нет, переход к блоку 31  
31. Переприсваивание переменных sequence++ = 8 + 1 = 9  
32. Вход в цикл: Переменная k = 4  
29. ‘а’ != ‘а’, нет, переход к блоку 31  
31. Переприсваивание переменных sequence++ = 9 + 1 = 10  
32. Вход в цикл: Переменная k = 5. Граница достигнута, переход к блоку 32  
32. 1 == 1, да, переход к блоку 33  
33. ‘ ‘ == ‘’ || ‘ ‘ == ‘\0’ || ‘ ‘ == ‘.’ || ‘ ‘ == ‘,’ || ‘ ‘ == ‘!’ || ‘ ‘ == ‘?’, да, переход к блоку 34  
34. 5 == 0 || ‘ ‘ == ‘ ‘, да, переход к блоку 36  
36. Возврат i из подпрограммы  
Переприсваивание переменных position = 5  
7. 5 != -1, да, переход к блоку 8  
8. Вход в цикл: Переменная i = 5  
9. ‘в‘ == ‘ ‘, нет, переход к блоку 8  
8. Вход в цикл: Переменная i = 4  
9. ‘ ‘ == ‘ ‘, да, переход к блоку 10  
10. Переприсваивание переменных counter++ = 1 + 1 = 2  
8. Вход в цикл: Переменная i = 3  
9. ‘а‘ == ‘ ‘, нет, переход к блоку 8  
8. Вход в цикл: Переменная i = 2  
9. ‘ш‘ == ‘ ‘, нет, переход к блоку 8  
8. Вход в цикл: Переменная i = 1  
9. ‘а‘ == ‘ ‘, нет, переход к блоку 8  
8. Вход в цикл: Переменная i = 0. Граница достигнута, переход к блоку 12.  
12. Вывод «Номер слова в заданном тексте: 2»  
13. Очистка динамической памяти у массивов main\_text, main\_word  
37. Конец  
  
Вариант II

1. Начало  
38. Переприсваивание переменных counter = 0  
2. Вывод «Введите текст: »  
3. Переход к выполнению подпрограммы read\_string():  
20. Работа с переменными:  
len = 1, выделяем память под s (1 байт), считываем введённым пользователем символ с консоли в переменную с = 'B'  
21. 'B' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (1 байт), s[0] = 'B', считываем в переменную с следующий символ с консоли 'i', len++ = 1+1=2  
21. 'i' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (2 байта), s[1] = 'i', считываем в переменную с следующий символ с консоли 'g', len++ = 2+1=3  
21. 'g' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (3 байта), s[2] = 'g', считываем в переменную с следующий символ с консоли ' ', len++ = 3+1=4  
21. ' ' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (4 байта), s[3] = ' ', считываем в переменную с следующий символ с консоли 'a', len++ = 4+1=5  
21. 'a' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (5 байтов), s[4] = 'a', считываем в переменную с следующий символ с консоли 'l', len++ = 5+1=6  
21. 'l' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (6 байтов), s[5] = 'l', считываем в переменную с следующий символ с консоли 'i', len++ = 6+1=7  
21. 'i' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (7 байтов), s[6] = 'i', считываем в переменную с следующий символ с консоли 'b', len++ = 7+1=8  
21. 'b' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (8 байтов), s[7] = 'b', считываем в переменную с следующий символ с консоли 'i', len++ = 8+1=9  
21. 'i' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (9 байтов), s[8] = 'i', считываем в переменную с следующий символ с консоли '.', len++ = 9+1=10  
21. '.' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (10 байтов), s[9] = '.', считываем в переменную с следующий символ с консоли '\n',   
len++ = 10 + 1 = 11  
21. '\n' != '\n', нет, переход к блоку 23  
23. Переприсваивание переменных s[10] = ‘\0’  
24. Возврат s из подпрограммы  
Переприсваивание переменных main\_text = “Big alibi.”  
4. Вывод «Введите ваше слово: »  
5. Переход к выполнению подпрограммы read\_string():  
20. Работа с переменными:  
len = 1, выделяем память под s (1 байт), считываем введённым пользователем символ с консоли в переменную с = 'b'  
21. 'b' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (1 байт), s[0] = 'b', считываем в переменную с следующий символ с консоли 'i', len++ = 1+1=2  
21. 'b' != '\n', да, переход к блоку 22  
22. Перевыделение памяти под символьный массив s (2 байта), s[1] = 'i', считываем в переменную с следующий символ с консоли '\n', len++ = 2+1=3  
21. '\n' != '\n', нет, переход к блоку 23  
23. Переприсваивание переменных s[2] = '\0'  
24. Возврат s из подпрограммы  
Переприсваивание переменных main\_word = “bi”  
6. Переход к выполнению подпрограммы find(main\_text, main\_word)  
25. Переход к выполнению подпрограммы format(text)  
14. Вход в цикл: Переменная k = 0  
15. ‘B’ >= ‘A’ && ‘B’ <= ‘Z’, да, переход к блоку 16  
16. Переприсваивание переменных something[0] = 'B' – 'A' + 'a' = 'b'  
17. ‘b’ >= ‘A’ && ‘b’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 1  
15. ‘i’ >= ‘A’ && ‘i’ <= ‘Z’, нет, переход к блоку 17  
17. ‘i’ >= ‘A’ && ‘i’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 2  
15. ‘g’ >= ‘A’ && ‘g’ <= ‘Z’, нет, переход к блоку 17  
17. ‘g’ >= ‘A’ && ‘g’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 3  
15. ‘ ’ >= ‘A’ && ‘ ’ <= ‘Z’, нет, переход к блоку 17  
17. ‘ ’ >= ‘A’ && ‘ ’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 4  
15. ‘a’ >= ‘A’ && ‘a’ <= ‘Z’, нет, переход к блоку 17  
17. ‘a’ >= ‘A’ && ‘a’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 5  
15. ‘l’ >= ‘A’ && ‘l’ <= ‘Z’, нет, переход к блоку 17  
17. ‘l’ >= ‘A’ && ‘l’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 6  
15. ‘i’ >= ‘A’ && ‘i’ <= ‘Z’, нет, переход к блоку 17  
17. ‘i’ >= ‘A’ && ‘i’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 7  
15. ‘b’ >= ‘A’ && ‘b’ <= ‘Z’, нет, переход к блоку 17  
17. ‘b’ >= ‘A’ && ‘b’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 8  
15. ‘i’ >= ‘A’ && ‘i’ <= ‘Z’, нет, переход к блоку 17  
17. ‘i’ >= ‘A’ && ‘i’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 9  
15. ‘.’ >= ‘A’ && ‘.’ <= ‘Z’, нет, переход к блоку 17  
17. ‘.’ >= ‘A’ && ‘.’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 10. Граница достигнута, переход к блоку 19  
19. Возврат something из подпрограммы  
Переприсваивание переменных text = “big alibi”  
Переходим к выполнению подпрограммы format(word):  
14. Вход в цикл: Переменная k = 0  
15. ‘b’ >= ‘A’ && ‘b’ <= ‘Z’, нет, переход к блоку 17  
17. ‘b’ >= ‘A’ && ‘b’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 1  
15. ‘i’ >= ‘A’ && ‘i’ <= ‘Z’, нет, переход к блоку 17  
17. ‘i’ >= ‘A’ && ‘i’ <= ‘Я’, нет, переход к блоку 14  
14. Вход в цикл: Переменная k = 2. Граница достигнута, переход к блоку 19  
19. Возврат something из подпрограммы  
Переприсваивание переменных word = “bi”  
26. Вход в цикл: Переменная i = 0  
27. Переприсваивание переменных fact = 1, sequence = 0  
28. Вход в цикл: Переменная k = 0  
29. 'b' != 'b', нет, переход в блок 31  
31. Переприсваивание переменных sequence++ = 0 + 1 = 1  
28. Вход в цикл: Переменная k = 1  
29. 'i' != 'i', нет, переход в блок 31  
31. Переприсваивание переменных sequence++ = 1 + 1 = 2  
28. Вход в цикл: Переменная k = 2. Граница достигнута, переход к блоку 32  
32. 1 == 1, да, переход к блоку 33  
33. 'g' == ' ' || 'g' == ' \0' || 'g' == '.', нет, переход к блоку 26  
26. Вход в цикл: Переменная i = 1  
27. Переприсваивание переменных fact = 1, sequence = 1  
28. Вход в цикл: Переменная k = 0  
29. 'b' != 'i', да, переход к блоку 30  
30. Переприсваивание переменных fact = 0  
32. 0 == 1, нет, переход к блоку 26  
26. Вход в цикл: Переменная i = 2  
27. Переприсваивание переменных fact = 1, sequence = 2  
28. Вход в цикл: Переменная k = 0  
29. 'b' != 'g', да, переход к блоку 30  
30. Переприсваивание переменных fact = 0  
32. 0 == 1, нет, переход к блоку 26  
26. Вход в цикл: Переменная i = 3  
27. Переприсваивание переменных fact = 1, sequence = 3  
28. Вход в цикл: Переменная k = 0  
29. 'b' != ' ', да, переход к блоку 30  
30. Переприсваивание переменных fact = 0  
32. 0 == 1, нет, переход к блоку 26  
26. Вход в цикл: Переменная i = 4  
27. Переприсваивание переменных fact = 1, sequence = 4  
28. Вход в цикл: Переменная k = 0  
29. 'b' != 'a', да, переход к блоку 30  
30. Переприсваивание переменных fact = 0  
32. 0 == 1, нет, переход к блоку 26  
26. Вход в цикл: Переменная i = 5  
27. Переприсваивание переменных fact = 1, sequence = 5  
28. Вход в цикл: Переменная k = 0  
29. 'b' != 'l', да, переход к блоку 30  
30. Переприсваивание переменных fact = 0  
32. 0 == 1, нет, переход к блоку 26  
26. Вход в цикл: Переменная i = 6  
27. Переприсваивание переменных fact = 1, sequence = 6  
28. Вход в цикл: Переменная k = 0  
29. 'b' != 'i', да, переход к блоку 30  
30. Переприсваивание переменных fact = 0  
32. 0 == 1, нет, переход к блоку 26  
26. Вход в цикл: Переменная i = 7  
27. Переприсваивание переменных fact = 1, sequence = 7  
28. Вход в цикл: Переменная k = 0  
29. 'b' != 'b', нет, переход в блок 31  
31. Переприсваивание переменных sequence++ = 7 + 1 = 8  
28. Вход в цикл: Переменная k = 1  
29. 'i' != 'i', нет, переход в блок 31  
31. Переприсваивание переменных sequence++ = 8 + 1 = 9  
28. Вход в цикл: Переменная k = 2. Граница достигнута, переход к блоку 32  
32. fact == 1, да, переход к блоку 33  
33. ‘.’ == ‘ ‘ || ‘.’ == ‘\0‘ || ‘.’ == ‘.‘ || ‘.’ == ‘,’ || ‘.’ == ‘!’ || ‘.’ == ‘?’, да, переход к блоку 34  
34. 7 == 0 || ‘i’ == ‘ ‘, нет, переход к блоку 26  
26. Вход в цикл: Переменная i = 8  
27. Переприсваивание переменных fact = 1, sequence = 8  
29. 'b' != 'i', да, переход к блоку 30  
30. Переприсваивание переменных fact = 0  
32. 0 == 1, нет, переход к блоку 26  
26. Вход в цикл: Переменная i = 9  
27. Переприсваивание переменных fact = 1, sequence = 9  
29. 'b' != '.', да, переход к блоку 30  
30. Переприсваивание переменных fact = 0  
32. 0 == 1, нет, переход к блоку 26  
26. Вход в цикл: Переменная i = 10. Граница достигнута, переход к блоку 35  
35. Возврат -1 из подпрограммы  
Переприсваивание переменных position = -1  
7. -1 != -1, нет, переход к блоку 11  
11. Вывод «Заданное слово в тексте отсутствует»  
13. Очистка динамической памяти у массивов main\_text, main\_word  
37. Конец

4. Код программы:

#include <iostream>

using namespace std;

//функция для чтения строки, задаваемой пользователем

char\* read\_string() {

int len = 1;

char\* s = (char\*)malloc(sizeof(char));

char c = getchar(); // считываем первый символ из консоли

while (c != '\n') {

s = (char\*)realloc(s, len \* sizeof(char)); // перераспределяем память

s[len - 1] = c;

c = getchar(); // и снова считываем символ из консоли

len++;

}

s[len - 1] = '\0';

return s;

}

//функция для преобразования всех букв строки в строчные

char\* format(char\* something) {

for (int i = 0; i < strlen(something); i++) {

if (something[i] >= 'A' && something[i] <= 'Z')

something[i] = something[i] + 'a' - 'A';

if (something[i] >= 'А' && something[i] <= 'Я')

something[i] = something[i] + 'а' - 'А';

}

return something;

}

//функция, отвечающая за поиск заданного слова в заданном тексте

int find(char\* text, char\* word) {

text = format(text); word = format(word); // для улучшенного поиска

for (int i = 0; i < strlen(text); i++) { // сделаем все символы строки

int fact = 1, sequence = i; // строго строчными

for (int k = 0; k < strlen(word); k++) {

if (word[k] != text[sequence]) {

fact = 0;

break;

}

sequence++;

}

if (fact == 1) { // цепочка проверок на независимость слова

if (text[sequence] == ' ' || text[sequence] == '\0' || text[sequence] == '.' || text[sequence] == ',' || text[sequence] == '!' || text[sequence] == '?') {

if (i == 0 || text[i - 1] == ' ')

return i;

}

}

}

return -1;

}

// точка входа в программу

int main() {

setlocale(LC\_ALL, "rus");

system("chcp 1251 >> null");

cout << "Введите текст: ";

char\* main\_text = read\_string();

cout << "Введите ваше слово: ";

char\* main\_word = read\_string();

int position = find(main\_text, main\_word), counter = 1;

if (position != -1) {

for (int i = position; i > 0; i--) {

if (main\_text[i] == ' ')

counter++; // подсчёт кол-ва пробелов перед словом

}

cout << "Номер слова в заданном тексте: " << counter << endl;

}

else

cout << "Заданное слово в тексте отсутствует." << endl;

main\_text = (char\*)realloc(NULL, 0); // освобождаем память

main\_word = (char\*)realloc(NULL, 0); // можно было бы через free,

system("pause"); // но так нагляднее!

return 0;

}

5. Результат выполнения работы программы:

Text

Description automatically generated

Отладочный пример №1

Text

Description automatically generated

Отладочный пример №2

6. Вывод:

В ходе выполнения работы были изучены способы задать программе массив символов произвольной длины, используя при этом функции распределения памяти malloc, realloc (calloc не употреблён по причине ненадобности). Также были изучены способы инициализации функции, которая возвращает не просто значение, но указатель на динамический массив со значениями.

Было проделано изучение материала по части использования так называемых 'высших функций', способов ветвления алгоритмов и обработки символов, также в ходе многострочной отладки были выявлены исключения, которые были успешно устранены на программном уровне.

На контрольных примерах мы убедились, что программа работает превосходно при указанных введённых данных.

Был оформлен комплект документации на программный код.

**Задание №10**

Задание: задан текст. Напечатать все слова, в которых повторяется первая буква слова.

1. Таблица имён:

|  |  |  |
| --- | --- | --- |
| Исходные данные | | |
| text | Указатель на символьный массив | Исходный текст, введённый пользователем |
| Рабочие переменные | | |
| temp\_text | Указатель на символьный массив | Копия исходного текста, чтобы при работе не испортить оригинал |
| text[i], text[p] | Символьный | Символ текста, введённого пользователем |
| temp\_text[i] temp\_text[i – 1] | Целочисленный | Символ копии текста, введённого пользователем |
| symbols | Указатель на символьный массив | Массив из всех первых букв слова в тексте, введённого пользователем |
| symbols[i] | Cимвольный | Первая буква i по счёту слова в text |
| symbols[k] | Cимвольный | Первая буква k по счёту слова в text |
| symbols\_am | Указатель на целочисленный массив | Массив, который показывает количество встреченных раз символа symbols[i]\symbols[k] в массиве symbols |
| arr | Указатель на символьный массив | Динамический массив для считывания текста, что задаёт пользователь |
| arr[len - 1] | Символьный | Новое место для записи символа с текста, заданного пользователем |
| len | Целочисленный | Постоянно меняющаяся длина динамического массива |
| с | Символьный | Символ, в который считывается весь текст, введённый пользователем |
| story | Указатель на символьный массив | Массив, в котором заглавные буквы преобразуются в строчные |
| spaces\_counter | Целочисленный | Счётчик пробелов в предложении |
| sentence | Указатель на символьный массив | Массив для подсчёта слов в тексте |
| wannaread | Указатель на символьный массив | Массив для анализа первых букв в слове в изначальном тексте пользователя |
| first | Указатель на символьный массив | Массив первых букв слов текста пользователя |
| first[k] | Символьный | Место, куда записывается первая буква слова из исходного текста |
| k | Целочисленный | Переменная-счётчик для заполнения массива first |
| k, i, p | Целочисленные | Переменные-счётчики в циклах |
| symbol | Символьный | Символ для поиска в тексте |
| text | Указатель на символьный массив | Текст, в котором ищется символ |
| text[k] | Символ | Символ, с которым сравнивается symbol для установления его позиции в тексте |
| m | Символьный | Символ текста для проверки слова на окончание одним из символов |
| Результат | | |
| Все такие text[p], при которых p лежит в отрезке от [0; s] или [i – 1; s] при text[i-1] – пробел, где s - позиция ближайшего символа конца слова в отрезке от [0; strlen(text) – 1] или [i-1; strlen(text) – 1] | | |

1. Diagram

   Description automatically generatedБлок-схема:
2. Отладочный пример:

Вариант I

1. Начало  
2. Переходим к выполнению подпрограммы reading():  
19. Вывод “Введите текст: ”  
20. Переприсваивание переменных  
len = 1, выделение памяти под динамический массив arr в размере (1), Считывание символа ‘М’ в переменную с  
21. ‘М’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти под динамический массив arr в размере (1)  
Переприсваивание переменных arr[0] = ‘M’, len++ = 1 + 1 = 2  
Считывание в переменную с символа ’а’  
21. ‘а’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти под динамический массив arr в размере (2)  
Переприсваивание переменных arr[1] = ‘а’, len++ = 2 + 1 = 3  
Считывание в переменную с символа ’м’  
21. ‘м’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти под динамический массив arr в размере (3)  
Переприсваивание переменных arr[2] = ‘м’, len++ = 3 + 1 = 4  
Считывание в переменную с символа ’а’  
21. ‘а’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти под динамический массив arr в размере (4)  
Переприсваивание переменных arr[3] = ‘а’, len++ = 4 + 1 = 5  
Считывание в переменную с символа ’,’  
21. ‘,’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти под динамический массив arr в размере (5)  
Переприсваивание переменных arr[4] = ‘,’, len++ = 5 + 1 = 6  
Считывание в переменную с символа ’ ’  
21. ‘ ’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти под динамический массив arr в размере (6)  
Переприсваивание переменных arr[5] = ‘ ’, len++ = 6 + 1 = 7  
Считывание в переменную с символа ’г’  
21. ‘г’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти под динамический массив arr в размере (7)  
Переприсваивание переменных arr[6] = ‘г’, len++ = 7 + 1 = 8  
Считывание в переменную с символа ’д’  
21. ‘д’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти под динамический массив arr в размере (8)  
Переприсваивание переменных arr[7] = ‘д’, len++ = 8 + 1 = 9  
Считывание в переменную с символа ’е’  
21. ‘е’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти под динамический массив arr в размере (9)  
Переприсваивание переменных arr[8] = ‘е’, len++ = 9 + 1 = 10  
Считывание в переменную с символа ’ ’  
21. ‘ ’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти под динамический массив arr в размере (10)  
Переприсваивание переменных arr[9] = ‘ ’, len++ = 10 + 1 = 11  
Считывание в переменную с символа ’о’  
21. ‘о’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти под динамический массив arr в размере (11)  
Переприсваивание переменных arr[10] = ‘о’, len++ = 11 + 1 = 12  
Считывание в переменную с символа ’ч’  
21. ‘ч’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти под динамический массив arr в размере (12)  
Переприсваивание переменных arr[11] = ‘ч’, len++ = 12 + 1 = 13  
Считывание в переменную с символа ’к’  
21. ‘к’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти под динамический массив arr в размере (13)  
Переприсваивание переменных arr[12] = ‘к’, len++ = 13 + 1 = 14  
Считывание в переменную с символа ’и’  
21. ‘и’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти под динамический массив arr в размере (14)  
Переприсваивание переменных arr[13] = ‘и’, len++ = 14 + 1 = 15  
Считывание в переменную с символа ’?’  
21. ‘?’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти под динамический массив arr в размере (15)  
Переприсваивание переменных arr[14] = ‘?’, len++ = 15 + 1 = 16  
Считывание в переменную с символа ’\n’  
21. ‘\n’ != ‘\n’, нет, переход к блоку 23  
23. Переприсваивание переменных arr[15] =’\0’  
24. Возврат ”Мама, где мои очки?\0” из подпрограммы  
Переприсваивание переменных text = ”Мама, где мои очки?”  
Выделение памяти под символьный массив temp\_text в размере (16)  
exception = true  
3. Вход в цикл: Переменная i = 0  
4. Переприсваивание переменных: temp\_text[0] = ‘М’  
3. Вход в цикл: Переменная i = 1  
4. Переприсваивание переменных: temp\_text[1] = ‘а’  
3. Вход в цикл: Переменная i = 2  
4. Переприсваивание переменных: temp\_text[2] = ‘м’  
3. Вход в цикл: Переменная i = 3  
4. Переприсваивание переменных: temp\_text[3] = ‘а’  
3. Вход в цикл: Переменная i = 4  
4. Переприсваивание переменных: temp\_text[4] = ‘,’  
3. Вход в цикл: Переменная i = 5  
4. Переприсваивание переменных: temp\_text[5] = ‘ ’  
3. Вход в цикл: Переменная i = 6  
4. Переприсваивание переменных: temp\_text[6] = ‘г’  
3. Вход в цикл: Переменная i = 7  
4. Переприсваивание переменных: temp\_text[7] = ‘д’  
3. Вход в цикл: Переменная i = 8  
4. Переприсваивание переменных: temp\_text[8] = ‘е’  
3. Вход в цикл: Переменная i = 9  
4. Переприсваивание переменных: temp\_text[9] = ‘ ’  
3. Вход в цикл: Переменная i = 10  
4. Переприсваивание переменных: temp\_text[10] = ‘о’  
3. Вход в цикл: Переменная i = 11  
4. Переприсваивание переменных: temp\_text[11] = ‘ч’  
3. Вход в цикл: Переменная i = 12  
4. Переприсваивание переменных: temp\_text[12] = ‘к’  
3. Вход в цикл: Переменная i = 13  
4. Переприсваивание переменных: temp\_text[13] = ‘и’  
3. Вход в цикл: Переменная i = 14  
4. Переприсваивание переменных: temp\_text[14] = ‘?’  
3. Вход в цикл: Переменная i = 15  
4. Переприсваивание переменных: temp\_text[14] = ‘\0’  
5. Переход к выполнению подпрограммы lower(temp\_text):  
25. Вход в цикл: Переменная i = 0  
26. ‘A’ <= ‘М‘ <= ‘Z’, нет, переход к блоку 28  
26. ‘A’ <= ‘М‘ <= ‘Я’, да, переход к блоку 29  
29. Переприсваивание переменных story[0] = ‘М‘ – ‘A’ + ‘a’ = ‘м’  
25. Вход в цикл: Переменная i = 1  
26. ‘A’ <= ‘а‘ <= ‘Z’, нет, переход к блоку 28  
26. ‘A’ <= ‘а‘ <= ‘Я’, нет, переход к блоку 25  
25. Вход в цикл: Переменная i = 2  
26. ‘A’ <= ‘м‘ <= ‘Z’, нет, переход к блоку 28  
26. ‘A’ <= ‘а‘ <= ‘Я’, нет, переход к блоку 25  
25. Вход в цикл: Переменная i = 3  
26. ‘A’ <= ‘а‘ <= ‘Z’, нет, переход к блоку 28  
26. ‘A’ <= ‘а‘ <= ‘Я’, нет, переход к блоку 25  
25. Вход в цикл: Переменная i = 4  
26. ‘A’ <= ‘,‘ <= ‘Z’, нет, переход к блоку 28  
26. ‘A’ <= ‘,‘ <= ‘Я’, нет, переход к блоку 25  
25. Вход в цикл: Переменная i = 5  
26. ‘A’ <= ‘ ‘ <= ‘Z’, нет, переход к блоку 28  
26. ‘A’ <= ‘ ‘ <= ‘Я’, нет, переход к блоку 25  
25. Вход в цикл: Переменная i = 6  
26. ‘A’ <= ‘г‘ <= ‘Z’, нет, переход к блоку 28  
26. ‘A’ <= ‘г‘ <= ‘Я’, нет, переход к блоку 25  
25. Вход в цикл: Переменная i = 7  
26. ‘A’ <= ‘д‘ <= ‘Z’, нет, переход к блоку 28  
26. ‘A’ <= ‘д‘ <= ‘Я’, нет, переход к блоку 25  
25. Вход в цикл: Переменная i = 8  
26. ‘A’ <= ‘е‘ <= ‘Z’, нет, переход к блоку 28  
26. ‘A’ <= ‘е‘ <= ‘Я’, нет, переход к блоку 25  
25. Вход в цикл: Переменная i = 9  
26. ‘A’ <= ‘ ‘ <= ‘Z’, нет, переход к блоку 28  
26. ‘A’ <= ‘ ‘ <= ‘Я’, нет, переход к блоку 25  
25. Вход в цикл: Переменная i = 10  
26. ‘A’ <= ‘о‘ <= ‘Z’, нет, переход к блоку 28  
26. ‘A’ <= ‘о‘ <= ‘Я’, нет, переход к блоку 25  
25. Вход в цикл: Переменная i = 11  
26. ‘A’ <= ‘ч‘ <= ‘Z’, нет, переход к блоку 28  
26. ‘A’ <= ‘ч‘ <= ‘Я’, нет, переход к блоку 25  
25. Вход в цикл: Переменная i = 12  
26. ‘A’ <= ‘к‘ <= ‘Z’, нет, переход к блоку 28  
26. ‘A’ <= ‘к‘ <= ‘Я’, нет, переход к блоку 25  
25. Вход в цикл: Переменная i = 13  
26. ‘A’ <= ‘и‘ <= ‘Z’, нет, переход к блоку 28  
26. ‘A’ <= ‘и‘ <= ‘Я’, нет, переход к блоку 25  
25. Вход в цикл: Переменная i = 14  
26. ‘A’ <= ‘?‘ <= ‘Z’, нет, переход к блоку 28  
26. ‘A’ <= ‘?‘ <= ‘Я’, нет, переход к блоку 25  
25. Вход в цикл: Переменная i = 15. Граница достигнута, переход в блок 30  
30. Возврат “мама, где очки?”  
Переприсваивание переменных   
temp\_text = lower(temp\_text) = “мама, где очки?\0”  
Переходим к выполнению подпрограммы first\_symbols(temp\_text):  
40. Переходим к выполнению подпрограммы words\_counter(wannaread):  
35. Переприсваивание переменных spaces\_counter = 1  
36. Вход в цикл: Переменная i = 0  
37. ‘м’ == ‘ ‘, нет, переход к блоку 36  
36. Вход в цикл: Переменная i = 1  
37. ‘а’ == ‘ ‘, нет, переход к блоку 36  
36. Вход в цикл: Переменная i = 2  
37. ‘м’ == ‘ ‘, нет, переход к блоку 36  
36. Вход в цикл: Переменная i = 3  
37. ‘а’ == ‘ ‘, нет, переход к блоку 36  
36. Вход в цикл: Переменная i = 4  
37. ‘,’ == ‘ ‘, нет, переход к блоку 36  
36. Вход в цикл: Переменная i = 5  
37. ‘ ’ == ‘ ‘, да, переход к блоку 38  
38. Переприсваивание переменных spaces\_counter++ = 1 + 1 = 2  
36. Вход в цикл: Переменная i = 6  
37. ‘г’ == ‘ ‘, нет, переход к блоку 36  
36. Вход в цикл: Переменная i = 7  
37. ‘д’ == ‘ ‘, нет, переход к блоку 36  
36. Вход в цикл: Переменная i = 8  
37. ‘е’ == ‘ ‘, нет, переход к блоку 36  
36. Вход в цикл: Переменная i = 9  
37. ‘ ’ == ‘ ‘, да, переход к блоку 38  
38. Переприсваивание переменных spaces\_counter++ = 2 + 1 = 3  
36. Вход в цикл: Переменная i = 10  
37. ‘о’ == ‘ ‘, нет, переход к блоку 36  
36. Вход в цикл: Переменная i = 11  
37. ‘ч’ == ‘ ‘, нет, переход к блоку 36  
36. Вход в цикл: Переменная i = 12  
37. ‘к’ == ‘ ‘, нет, переход к блоку 36  
36. Вход в цикл: Переменная i = 13  
37. ‘и’ == ‘ ‘, нет, переход к блоку 36  
36. Вход в цикл: Переменная i = 14  
37. ‘?’ == ‘ ‘, нет, переход к блоку 36  
36. Вход в цикл: Переменная i = 15  
37. ‘\0’ == ‘ ‘, нет, переход к блоку 36  
36. Вход в цикл: Переменная i = 16. Граница достигнута, переход к блоку 39  
39. Возврат 3 из подпрограммы  
Переприсваивание переменных   
k = 0, words = words\_counter(wannaread) = 3  
Выделение памяти под символьный массив first в размере 3  
41. Вход в цикл: Переменная i = 0  
42. 0 == 0, да, переход в блок 43  
43. Переприсваивание переменных first[0] = 'м’, k++ = 0 + 1 = 1  
41. Вход в цикл: Переменная i = 1  
42. 1 == 0, нет, переход в блок 44  
44. 'м' == ' ', нет, переход в блок 45  
45. 1 == 3, нет, переход к блоку 41  
41. Вход в цикл: Переменная i = 2  
42. 2 == 0, нет, переход в блок 44  
44. 'а' == ' ', нет, переход в блок 45  
45. 1 == 3, нет, переход к блоку 41  
41. Вход в цикл: Переменная i = 3  
42. 3 == 0, нет, переход в блок 44  
44. 'м' == ' ', нет, переход в блок 45  
45. 1 == 3, нет, переход к блоку 41  
41. Вход в цикл: Переменная i = 4  
42. 4 == 0, нет, переход в блок 44  
44. 'а' == ' ', нет, переход в блок 45  
45. 1 == 3, нет, переход к блоку 41  
41. Вход в цикл: Переменная i = 5  
42. 5 == 0, нет, переход в блок 44  
44. ',' == ' ', нет, переход в блок 45  
45. 1 == 3, нет, переход к блоку 41  
41. Вход в цикл: Переменная i = 6  
42. 6 == 0, нет, переход в блок 44  
44. ' ' == ' ', да, переход в блок 43  
43. Переприсваивание переменных first[1] = 'г’, k++ = 1 + 1 = 2  
41. Вход в цикл: Переменная i = 7  
42. 7 == 0, нет, переход в блок 44  
44. 'г' == ' ', нет, переход в блок 45  
45. 2 == 3, нет, переход к блоку 41  
41. Вход в цикл: Переменная i = 8  
42. 8 == 0, нет, переход в блок 44  
44. 'д' == ' ', нет, переход в блок 45  
45. 2 == 3, нет, переход к блоку 41  
41. Вход в цикл: Переменная i = 8  
42. 8 == 0, нет, переход в блок 44  
44. 'е' == ' ', нет, переход в блок 45  
45. 2 == 3, нет, переход к блоку 41  
41. Вход в цикл: Переменная i = 9  
42. 9 == 0, нет, переход в блок 44  
44. ' ' == ' ', да, переход в блок 43  
43. Переприсваивание переменных first[2] = 'о’, k++ = 2 + 1 = 3  
41. Вход в цикл: Переменная i = 10  
42. 10 == 0, нет, переход в блок 44  
44. 'о' == ' ', нет, переход в блок 45  
45. 3 == 3, да, переход в блок 46  
46. Ссылка на блок 47  
47. Возврат “мго” из подпрограммы  
Переприсваивание переменных symbols = first\_symbols(temp\_text) = “мго”   
Выделение памяти под целочисленный массив symbols\_am в размере (16)  
6. Вход в цикл: Переменная i = 0  
7. Вход в цикл: Переменная k = 0  
8. 'м' == 'м', да, переход к блоку 9  
9. symbols\_am[0] = 0 + 1 = 1  
7. Вход в цикл: Переменная k = 1  
8. 'г' == 'м', нет, переход к блоку 7  
7. Вход в цикл: Переменная k = 2  
8. 'о' == 'м', нет, переход к блоку 7  
7. Вход в цикл: Переменная k = 3. Граница достигнута, переход к блоку 6  
6. Вход в цикл: Переменная i = 1  
7. Вход в цикл: Переменная k = 0  
8. 'м' == 'г', нет, переход к блоку 7  
7. Вход в цикл: Переменная k = 1  
8. 'г' == 'г', да, переход к блоку 9  
9. symbols\_am[1] = 0 + 1 = 1  
7. Вход в цикл: Переменная k = 2  
8. 'о' == 'г', нет, переход к блоку 7  
7. Вход в цикл: Переменная k = 3. Граница достигнута, переход к блоку 6  
6. Вход в цикл: Переменная i = 2  
7. Вход в цикл: Переменная k = 0  
8. 'м' == 'о', нет, переход к блоку 7  
7. Вход в цикл: Переменная k = 1  
8. 'г' == 'о', нет, переход к блоку 7  
7. Вход в цикл: Переменная k = 2  
8. 'о' == 'о', да, переход к блоку 9  
 9. symbols\_am[2] = 0 + 1 = 1  
6. Вход в цикл: Переменная i = 3. Граница достигнута, переход к блоку 10  
10. Вывод (перенос на следующую строку)“Ответ: ”  
11. Вход в цикл: Переменная i = 0  
12. 0 == 0 || ‘{Some garbage}’ == ‘ ’, да, переход к блоку 13  
13. Переходим к выполнению подпрограммы index(temp\_text[0], symbols)  
31. Вход в цикл: Переменная k = 0  
32. 'м' == 'м', да, переход к блоку 33  
33. Возврат 0 из подпрограммы  
1 > 1, нет, переход к блоку 11  
11. Вход в цикл: Переменная i = 1  
12. 1 == 0 || ‘м’ == ‘ ’, нет, переход к блоку 11  
11. Вход в цикл: Переменная i = 2  
12. 2 == 0 || ‘а’ == ‘ ’, нет, переход к блоку 11  
11. Вход в цикл: Переменная i = 3  
12. 3 == 0 || ‘м’ == ‘ ’, нет, переход к блоку 11  
11. Вход в цикл: Переменная i = 4  
12. 4 == 0 || ‘а’ == ‘ ’, нет, переход к блоку 11  
11. Вход в цикл: Переменная i = 5  
12. 5 == 0 || ‘,’ == ‘ ’, нет, переход к блоку 11  
11. Вход в цикл: Переменная i = 6  
12. 6 == 0 || ‘ ’ == ‘ ’, да, переход к блоку 13  
13. Переходим к выполнению подпрограммы index(temp\_text[6], symbols)  
31. Вход в цикл: Переменная k = 0  
32. 'м' == 'г', нет, переход к блоку 31  
31. Вход в цикл: Переменная k = 1  
32. 'г' == 'г', да, переход к блоку 33  
33. Возврат 1 из подпрограммы  
1 > 1, нет, переход к блоку 11  
11. Вход в цикл: Переменная i = 7  
12. 9 == 0 || ‘г’ == ‘ ’, нет, переход к блоку 11  
11. Вход в цикл: Переменная i = 8  
12. 8 == 0 || ‘д’ == ‘ ’, нет, переход к блоку 11  
11. Вход в цикл: Переменная i = 9  
12. 9 == 0 || ‘е’ == ‘ ’, нет, переход к блоку 11  
11. Вход в цикл: Переменная i = 10  
12. 10 == 0 || ‘ ’ == ‘ ’, да, переход к блоку 13  
13. Переходим к выполнению подпрограммы index(temp\_text[10], symbols)  
31. Вход в цикл: Переменная k = 0  
32. 'м' == 'о', нет, переход к блоку 31  
31. Вход в цикл: Переменная k = 1  
32. 'г' == 'о', нет, переход к блоку 31  
31. Вход в цикл: Переменная k = 2  
32. 'о' == 'о', да, переход к блоку 33  
33. Возврат 2 из подпрограммы  
1 > 1, нет, переход к блоку 11  
11. Вход в цикл: Переменная i = 11  
12. 11 == 0 || ‘о’ == ‘ ’, нет, переход к блоку 11  
11. Вход в цикл: Переменная i = 12  
12. 12 == 0 || ‘ч’ == ‘ ’, нет, переход к блоку 11  
11. Вход в цикл: Переменная i = 13  
12. 13 == 0 || ‘к’ == ‘ ’, нет, переход к блоку 11  
11. Вход в цикл: Переменная i = 14  
12. 14 == 0 || ‘и’ == ‘ ’, нет, переход к блоку 11  
11. Вход в цикл: Переменная i = 15. Граница достигнута, переход к блоку 51  
52. True, да, переходим к блоку 52  
53. Переходим к выполнению подпрограммы index(‘s’, (char\*)”k”):  
31. Вход в цикл: Переменная k = 0  
32. ‘s’ == ‘k’, нет, переход к блоку 31  
31. Вход в цикл: Переменная k = 1. Граница достигнута, переходим в блок 34  
34. Возврат -1 из подпрограммы  
Вывод “В данном тексте нет слов с одинаковыми буквами. (Код ошибки: -1)”  
54. Перевод текста на новую строку  
55. Конец

Вариант II

1. Начало  
2. Переходим к выполнению подпрограммы reading():  
19. Вывод “Введите текст: ”  
20. Переприсваивание переменных len = 1  
Выделение памяти под динамический массив arr в размере (1)  
Считывание символа ‘I’ из консоли в переменную с  
21. ‘I’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти в динамический массив arr в размере (1)  
Переприсваивание переменных arr[0] = ‘I’, len++ = 1 + 1 = 2  
Считывание символа ‘ ’ из консоли в переменную с  
21. ‘ ’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти в динамический массив arr в размере (2)  
Переприсваивание переменных arr[1] = ‘I’, len++ = 2 + 1 = 3  
Считывание символа ‘i’ из консоли в переменную с  
21. ‘i’ != ‘\n’, да, переход к блоку 22  
22. Перевыделение памяти в динамический массив arr в размере (3)  
Переприсваивание переменных arr[2] = ‘i’, len++ = 3 + 1 = 4  
Считывание символа ‘\n’ из консоли в переменную с  
21. ‘\n’ != ‘\n’, нет, переход к блоку 23  
23. Переприсваивание переменных arr[3] = ‘\0’  
24. Возврат “I i” из подпрограммы  
Переприсваивание переменных: text = “I i”, exception = true  
Выделение памяти для символьного массива temp\_text в размере (4)  
3. Вход в цикл: Переменная i = 0  
4. Переприсваивание переменных temp\_text[0] = ‘I’  
3. Вход в цикл: Переменная i = 1  
4. Переприсваивание переменных temp\_text[1] = ‘ ’  
3. Вход в цикл: Переменная i = 2  
4. Переприсваивание переменных temp\_text[2] = ‘i’  
3. Вход в цикл: Переменная i = 3. Граница достигнута, переход к блоку 5  
5. Переходим к выполнению подпрограммы lower(temp\_text)   
25. Вход в цикл: Переменная i = 0  
26. 'A' <= 'I' <= 'Z', да, переход к блоку 27  
27. story[0] = 'I' – 'A' + 'a’ = ‘i’  
28. ‘A’ <= ‘i' <= ‘Я’, нет, переход к блоку 25  
25. Вход в цикл: Переменная i = 1  
26. 'A' <= ' ' <= 'Z', нет, переход к блоку 28  
28. ‘A’ <= ‘ ' <= ‘Я’, нет, переход к блоку 25  
25. Вход в цикл: Переменная i = 2  
26. 'A' <= 'i' <= 'Z', нет, переход к блоку 28  
28. ‘A’ <= ‘i' <= ‘Я’, нет, переход к блоку 25  
25. Вход в цикл: Переменная i = 3. Граница достигнута, переход к блоку 30  
30. Возврат “i i” из подпрограммы  
Переходим к выполнению подпрограммы first\_symbols(temp\_text):  
40. Переходим к выполнению подпрограммы words\_couter(wannaread)  
35. Переприсваивание переменных spaces\_counter = 1  
36. Вход в цикл: Переменная i = 0  
37. 'i' == ' ', нет, переход к блоку 36  
36. Вход в цикл: Переменная i = 1  
37. ' ' == ' ', да, переход к блоку 38  
38. Переприсваивание переменных spaces\_counter++ = 1 + 1 = 2  
36. Вход в цикл: Переменная i = 2  
37. 'i' == ' ', нет, переход к блоку 36  
36. Вход в цикл: Переменная i = 3. Граница достигнута, переход к блоку 39  
39. Возврат 2 из подпрограммы  
Переприсваивание переменных k = 0, words = 2  
Выделение памяти под символьный массив first в размере (2)  
41. Вход в цикл: Переменная i = 0  
42. 0 == 0, да, переход к блоку 43  
43. Переприсваивание переменных first[0] = 'i', k++ = 0 + 1 = 1  
41. Вход в цикл: Переменная i = 1  
42. 1 == 0, нет, переход к блоку 44  
44. 'i' == ' ', нет, переход к блоку 45  
45. 1 == 2, нет, переход к блоку 41  
41. Вход в цикл: Переменная i = 2  
42. 2 == 0, нет, переход к блоку 44  
44. ' ' == ' ', да, переход к блоку 43  
43. Переприсваивание переменных first[1] = 'i', k++ = 1 + 1 = 2  
41. Вход в цикл: Переменная i = 3. Граница достигнута , переход к блоку 47  
47. Переприсваивание переменных first[2] = '\0'  
48. Возврат “ii” из подпрограммы  
Переприсваивание переменных temp\_text = lower(temp\_text) = “i i”,   
symbols = first\_symbols(temp\_text) = “ii”  
Выделение памяти для целочисленного массива symbols\_am и мгновенная инициализация его элементов в 0, в размере (2)  
6. Вход в цикл: Переменная i = 0  
  
7. Вход в цикл: Переменная k = 0  
8. 'i' == 'i', да, переход к блоку 9  
9. Переприсваивание переменных symbols\_am[0]++ = 0 + 1 = 1  
7. Вход в цикл: Переменная k = 1  
8. 'i' == 'i', да, переход к блоку 9  
9. Переприсваивание переменных symbols\_am[0]++ = 1 + 1 = 2  
7. Вход в цикл: Переменная k = 2. Граница достигнута, переход к блоку 6  
6. Вход в цикл: Переменная i = 1  
7. Вход в цикл: Переменная k = 0  
8. 'i' == 'i', да, переход к блоку 9  
9. Переприсваивание переменных symbols\_am[1]++ = 0 + 1 = 1  
7. Вход в цикл: Переменная k = 1  
8. 'i' == 'i', да, переход к блоку 9  
9. Переприсваивание переменных symbols\_am[1]++ = 1 + 1 = 2  
7. Вход в цикл: Переменная k = 2. Граница достигнута, переход к блоку 6  
6. Вход в цикл: Переменная i = 2. Граница достигнута, переход к блоку 10  
10. Перенос текста на новую строку, вывод “Ответ” на экран  
11. Вход в цикл: Переменная i = 0  
12. 0 == 0 || <Some\_garbage> == ' ', да, переход к блоку 13  
13. Переходим к выполнению подпрограммы index(temp\_text[0], symbols)  
31. Вход в цикл: Переменная k = 0  
32. 'i' == 'i', да, переход к блоку 33  
33. Возврат 0 из подпрограммы  
2 > 1, да, переход к блоку 14  
14. Переприсваивание переменных exception = false  
15. Вход в цикл: Переменная p = 0  
16. Переходим к выполнению подпрограммы end(text[p])  
49. 'I' == '.' || 'I' == ' ' || 'I' == '?' || 'I' == '!' || 'I' == ',', нет, переход к блоку 51  
51. Возврат false из подпрограммы  
false, нет, переход к блоку 17  
17. Вывод ‘I’ на экран  
15. Вход в цикл: Переменная p = 1  
16. Переходим к выполнению подпрограммы end(text[p])  
49. ' ' == '.' || ' ' == ' ' || ' ' == '?' || ' ' == '!' || ' ' == ',', да, переход к блоку 50  
50. Возврат true из подпрограммы  
true, да, переход к блоку 18  
18. Вывод пробела на экран  
11. Вход в цикл: Переменная i = 1  
12. 1 == 0 || 'I' == ' ', нет, переход к блоку 11  
11. Вход в цикл: Переменная i = 2  
12. 2 == 0 || ' ' == ' ', да, переход к блоку 13  
13. Переход к выполнению подпрограммы index(temp\_text[i], symbols):  
31. Вход в цикл: Переменная k = 0  
32. 'i' == 'i', да, переход к блоку 33  
33. Возврат 0 из подпрограммы  
2 > 1, да, переход к блоку 14  
14. Переприсваивание переменных exception = false  
15. Вход в цикл: Переменная p = 2  
16. Переходим к выполнению подпрограммы end(text[p])  
49. 'i' == '.' || 'i' == ' ' || 'i' == '?' || 'i' == '!' || 'i' == ',', нет, переход к блоку 51  
51. Возврат false из подпрограммы  
false, нет, переход к блоку 17  
17. Вывод ‘I’ на экран  
15. Вход в цикл: Переменная p = 3. Граница достигнута, переход к блоку 11  
11. Вход в цикл: Переменная i = 3. Граница достигнута, переход к блоку 52  
52. false, нет, переход к блоку 54  
54. Перевод слова на новую строку  
55. Конец

1. Код программы:

#include <iostream>

using namespace std;

// Читаем строку, введённую пользователем

char\* reading() {

cout << "Введите текст:" << " ";

char\* arr = (char\*)malloc(sizeof(char));

int len = 1; char c = getchar();

while (c != '\n') {

arr = (char\*)realloc(arr, len \* sizeof(char));

arr[len - 1] = c;

len++; c = getchar();

}

arr[len - 1] = '\0';

return arr;

}

// Для преобразования заглавных букв в строчные

char\* lower(char\* story) {

for (int i = 0; i < strlen(story); i++) {

if (story[i] >= 'A' && story[i] <= 'Z')

story[i] = story[i] - 'A' + 'a';

if (story[i] >= 'А' && story[i] <= 'Я')

story[i] = story[i] - 'А' + 'а';

}

return story;

}

// Считаем количество слов в предложении

int words\_counter(char\* sentence) {

int spaces\_counter = 1;

for (int i = 0; i < strlen(sentence); i++) {

if (sentence[i] == ' ')

spaces\_counter++;

}

return spaces\_counter;

}

// Читаем все первые символы в строке

char\* first\_symbols(char\* wannaread) {

int k = 0, words = words\_counter(wannaread);

char\* first = (char\*)malloc(words \* sizeof(char));

for (int i = 0; i < strlen(wannaread); i++) {

if (i == 0) {

first[k] = wannaread[i];

k++;

}

else if (wannaread[i - 1] == ' ') {

first[k] = wannaread[i];

k++;

}

else if (k == words) {

first[k] = '\0';

break;

}

}

return first;

}

// Узнаем номер позиции - первого вхождения заданного символа в строку

int index(char symbol, char\* text) {

for (int k = 0; k < strlen(text); k++) {

if (text[k] == symbol)

return k;

}

return -1;

}

// Проверяем, правда ли слово закончилось на один из этих символов

bool end(char m) {

if (m == '.' || m == ' ' || m == '?' || m == '!' || m == ',')

return true;

return false;

}

int main() {

setlocale(LC\_ALL, "rus");

system("chcp 1251 >> null");

char\* text = reading();

bool exception = true;

// Выделяем память в другой ячейке памяти для копии текста

char\* temp\_text = (char\*)malloc(strlen(text) \* sizeof(char));

// Работаем с копией, чтобы не портить исходный

for (int i = 0; i < strlen(text); i++)

temp\_text[i] = text[i];

// Копию текста понижаем для поиска одинаковых букв в разных регистрах

temp\_text = lower(temp\_text);

// Берём все первые символы в строке

char\* symbols = first\_symbols(temp\_text);

// Делаем аналогию словаря из двух массивов - symbols, symbols\_am, где соответственно находится буква и её количество среди первых букв слов в тексте

int\* symbols\_am = (int\*)calloc(strlen(symbols), sizeof(int));

for (int i = 0; i < strlen(symbols); i++) {

for (int k = 0; k < strlen(symbols); k++) {

// Для каждой буковки проверяем её количество в массиве symbols

if (symbols[k] == symbols[i])

symbols\_am[i]++;

}

}

cout << endl << "Ответ: ";

for (int i = 0; i < strlen(text); i++) {

// Проверяем, точно ли это слово с левой стороны

if (i == 0 || temp\_text[i - 1] == ' ') {

// Если первая буква слова встречается больше одного раза

if (symbols\_am[index(temp\_text[i], symbols)] > 1) {

exception = false;

for (int p = i; p < strlen(text); p++) {

// Пока не встретим символ завершения слова, печатаем побуквенно слово.

if (end(text[p])) {

cout << " ";

break;

}

cout << text[p];

}

}

}

}

if (exception)

printf("В заданном тексте нет слов с одинаковыми первыми буквами. (Код ошибки: %d)", index('s', (char\*)"k"));  
 // Для красоты вывода.

cout << endl;

return 0;

}

1. Результат выполнения работы программы:

Text

Description automatically generated

Отладочный пример №1  
Text

Description automatically generated

Отладочный пример №2

1. Вывод:

В ходе выполнения работы были изучены способы получения текста, задаваемого пользователем через консольную строку в символьный массив через работу с указателями. Изучили способы мгновенной инициализации элементов целочисленного массива при выделении ему памяти через функцию calloc(). Были изучены словари, и на основе их идеи были созданы два парных массива – symbols & symbols\_am, первый содержал в себе первые буквы всех слов, а второй на таких же позициях размещал количество таких букв среди всех таких же в первом массиве. Задание выполнялось в несколько этапов – понимание смысла задачи, быстрый план по пунктам на листочке бумаги, формирование блок-схемы и кода, шлифовка кода, подготовка отчёта. Функции названы мнемонично их функционалу.

Было проделано тщательное изучение материалов, посвящённых с/с++ и работе с этими языками, в том числе книга Т.А. Павловской (Программирование на языке высокого уровня)

На контрольных примерах мы убедились, что наш код работает исправно.

Был оформлен комплект документации на программный код.