|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** \_***ИУК «Информатика и управление»*\_\_**\_\_\_\_\_\_\_\_\_\_\_\_

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА**

**«Действия над одномерными массивами и строками»**

**ДИСЦИПЛИНА: «Высокоуровневое программирование»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-12Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) (Ф.И.О.) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_Пчелинцева Н.И. )  (Подпись) (Ф.И.О.) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |
| Калуга, 2020 | | |

**Цель:** изучение приемов работы с одномерными массивами и строковыми типами данных средствами языка С++.

**Задачи:** 1. Разработать программы с использованием массивов. 2. Познакомиться с операциями обработки элементов в массивах. 3. Изучить типовые алгоритмы решения задач с использованием массивов. 4. Научиться представлять сроку в С++ как массив символов. 5. Изучить основные функции и процедуры обработки строкового типа данных.

**Вариант 2**

**Задача 1.1**

Указать натуральное число из заданного интервала, в двоичном представлении которого больше всего единиц.

**Блок-схема:**





**Листинг:**

#include <iostream>

using namespace std;

int findmaxnumofunits(int beg, int end);

void main()

{

setlocale(LC\_ALL, "Russian");

int beg{};

cout << "Введите начало интервала: ";

cin >> beg;

int end{};

cout << "Введите конец интервала: ";

cin >> end;

cout << "Число с максимальным количеством единиц"

<< endl <<"в двоичном представлении: "

<< findmaxnumofunits(beg, end) << endl;

}

int findmaxnumofunits(int beg, int end)

{

int maxnum = 0;

int maxunit = -1;

for (int i = beg; i < end; i++)

{

int num = i;

int numofunit = 0;

while (num != 0)

{

if (num % 2 == 1)

{

numofunit++;

}

num = num / 2;

}

if (numofunit >= maxunit)

{

maxnum = i;

maxunit = numofunit;

}

}

return maxnum;

}

**Тест:**

Введите начало интервала: 5

Введите конец интервала: 13

Число с максимальным количеством единиц

в двоичном представлении: 11

**Задача 1.2**

Переформировать матрицу таким образом, чтобы ее столбцы располагались по убыванию их поэлементных сумм.

**Блок-схема:**







**Листинг:**

#include <iostream>

#include <iomanip>

using namespace std;

const int MAXLEN = 20;

void myswap(int& a, int& b);

int numlen(int num);

void sort(int sums[], int indexes[], int matrix\_columns);

void main()

{

setlocale(LC\_ALL, "Russian");

int matrix\_lines{};

cout << "Введите кол-во строк матрицы: ";

cin >> matrix\_lines;

int matrix\_columns{};

cout << "Введите кол-во столбцов матрицы: ";

cin >> matrix\_columns;

int maxlen = 1;

int Matrix[MAXLEN][MAXLEN]{};

cout << "Введите матрицу: " << endl;

for (int i = 0; i < matrix\_lines; i++)

{

for (int j = 0; j < matrix\_columns; j++)

{

cin >> Matrix[i][j];

int len = numlen(Matrix[i][j]);

if (len > maxlen)

{

maxlen = len;

}

}

}

int sums[MAXLEN]{};

int indexes[MAXLEN]{};

for (int m = 0; m < matrix\_columns; m++)

{

indexes[m] = m;

for (int n = 0; n < matrix\_lines; n++)

{

sums[m] += Matrix[n][m];

}

}

sort(sums, indexes, matrix\_columns);

cout << "Отсортированная матрица: " << endl;

for (int a = 0; a < matrix\_lines; a++)

{

int TempMatrix[MAXLEN]{};

for (int b = 0; b < matrix\_columns; b++)

{

TempMatrix[b] = Matrix[a][b];

}

for (int c = 0; c < matrix\_columns; c++)

{

Matrix[a][c] = TempMatrix[indexes[c]];

cout << setw(maxlen + 1) << Matrix[a][c];

}

cout << endl;

}

}

void myswap(int& a, int& b)

{

int temp = a;

a = b;

b = temp;

}

int numlen(int num)

{

int len = 1;

if (num < 0)

{

len++;

num = abs(num);

}

while (num / 10 != 0)

{

num = num / 10;

len++;

}

return len;

}

void sort(int sums[], int indexes[], int matrix\_columns)

{

bool check = true;

int start = 0;

int end = matrix\_columns - 1;

while (check)

{

check = false;

for (int p = start; p < end; p++)

{

if (sums[p] < sums[p + 1])

{

check = true;

myswap(sums[p], sums[p + 1]);

myswap(indexes[p], indexes[p + 1]);

}

}

start++;

for (int q = end; q > start - 1; q--)

{

if (sums[q] > sums[q - 1])

{

check = true;

myswap(sums[q], sums[q - 1]);

myswap(indexes[q], indexes[q - 1]);

}

}

end--;

}

}

**Тест:**

Введите кол-во строк матрицы: 2

Введите кол-во столбцов матрицы: 3

Введите матрицу:

10 4 1

2 12 6

Отсортированная матрица:

4 10 1

12 2 6

**Задача 1.3**

Используя метод “Решето Эратосфена”, по заданному натуральному N>1 найти все простые числа, меньшие N.

**Блок-схема:**







**Листинг:**

#include <iostream>

using namespace std;

const int MAXLEN = 255;

void filling(int Digits[], int border);

void piercing(int Digits[], int border);

void main()

{

setlocale(LC\_ALL, "Russian");

cout << "Введите натуральное число N>1: ";

int N{};

cin >> N;

int Digits[MAXLEN]{};

filling(Digits, N);

piercing(Digits, N);

cout << "Простые числа до N: " << endl;

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

{

if (Digits[i] != 0)

{

cout << Digits[i] << " ";

}

}

cout << endl;

}

void filling(int Digits[], int border)

{

for (int d = 2; d < border; d++)

{

Digits[d - 2] = d;

}

}

void piercing(int Digits[], int border)

{

int curnum = 2;

int num4check = 3;

while (curnum < border)

{

for (int n4c = num4check; n4c < border; n4c++)

{

if (n4c % curnum == 0 && Digits[n4c - 2] != 0)

{

Digits[n4c - 2] = 0;

}

}

curnum++;

num4check++;

while (Digits[curnum - 2] == 0 && curnum < border)

{

curnum++;

num4check++;

}

}

}

**Тест:**

Введите натуральное число N>1: 28

Простые числа до N:

2 3 5 7 11 13 17 19 23

**Задача 1.4**

Из заданного на плоскости множества точек выбрать такие три, которые составляют треугольник наибольшего периметра.

**Блок-схема:**







**Листинг:**

#include <iostream>

using namespace std;

const int MAXLEN = 255;

float perimeter(float Ax, float Ay, float Bx, float By,

float Cx, float Cy);

void main()

{

setlocale(LC\_ALL, "Russian");

cout << "Введите кол-во точек: ";

int numofpoints{};

cin >> numofpoints;

cout << "Введите координаты точек:" << endl;

float Points[MAXLEN][2]{};

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

{

cin >> Points[i][0];

cin >> Points[i][1];

}

int p1 = 0;

int p2 = 1;

int p3 = 2;

int maxper = 0;

for (int a = 0; a < numofpoints - 2; a++)

{

for (int b = a + 1; b < numofpoints - 1; b++)

{

for (int c = b + 1; c < numofpoints; c++)

{

float per = perimeter(Points[a][0],

Points[a][1], Points[b][0],

Points[b][1], Points[c][0],

Points[c][1]);

if (per > maxper)

{

maxper = per;

p1 = a;

p2 = b;

p3 = c;

}

}

}

}

cout << "Точки, образующие треугольник с наибольшим периметром: " << endl

<< Points[p1][0] << " " << Points[p1][1] << endl

<< Points[p2][0] << " " << Points[p2][1] << endl

<< Points[p3][0] << " " << Points[p3][1] << endl;

}

float perimeter(float Ax, float Ay, float Bx, float By, float Cx, float Cy)

{

return sqrt(pow(Bx - Ax, 2) + pow(By - Ay, 2))

+ sqrt(pow(Cx - Bx, 2) + pow(Cy - By, 2))

+ sqrt(pow(Ax - Cx, 2) + pow(Ay - Cy, 2));

}

**Тест:**

Введите кол-во точек: 4

Введите координаты точек:

0 0

3 0

0 4

0 6

Точки, образующие треугольник с наибольшим периметром:

0 0

3 0

0 6

**Задача 1.5**

Выпуклый многоугольник задан координатами своих вершин. Вершины упорядочены. Вычислить площадь многоугольника.

**Блок-схема:**



**Листинг:**

#include <iostream>

using namespace std;

const int MAXLEN = 255;

float square(float Ax, float Ay, float Bx,

float By, float Cx, float Cy);

void main()

{

setlocale(LC\_ALL, "Russian");

cout << "Введите вол-во точек: ";

int nop{};

cin >> nop;

cout << "Введите координаты многоугоольника: " << endl;

float Points[MAXLEN][2]{};

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

{

cin >> Points[i][0];

cin >> Points[i][1];

}

float sq = 0;

for (int j = 0; j < nop - 2; j++)

{

sq += square(Points[0][0], Points[0][1],

Points[j + 1][0], Points[j + 1][1],

Points[j + 2][0], Points[j + 2][1]);

}

cout << "Площадь фигуры: " << sq << endl;

}

float square(float Ax, float Ay, float Bx,

float By, float Cx, float Cy)

{

float ABx = Bx - Ax;

float ABy = By - Ay;

float ACx = Cx - Ax;

float ACy = Cy - Ay;

return 0.5 \* abs(ABx \* ACy - (ABy \* ACx));

}

**Тест:**

Введите вол-во точек: 4

Введите координаты многоугоольника:

1 0

6 0

8 3

3 3

Площадь фигуры: 15

**Задача 2.1**

Ввести строку, состоящую только из цифр и букв. Распечатать те группы цифр, в которых цифра 7 встречается не более двух раз. (Группа цифр - это последовательность цифр, обрамленная буквами.)

**Блок-схема:**









**Листинг:**

#include <iostream>

using namespace std;

const int MAXLEN = 255;

void clear(char word[], int len);

void main()

{

setlocale(LC\_ALL, "Russian");

cout << "Введите строку: " << endl;

char str[MAXLEN]{};

cin.getline(str, MAXLEN, '\n');

int len = strlen(str);

char num[MAXLEN]{};

int n = 0;

int seven = 0;

bool exist = false;

cout << "Числа, содержащие не более двух 7: " << endl;

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

{

if (isdigit(str[i]))

{

if (str[i] == '7')

{

seven++;

}

num[n] = str[i];

n++;

if (isalpha(str[i + 1]))

{

if (seven < 3)

{

exist = true;

cout << num << " ";

}

seven = 0;

clear(num, len);

n = 0;

}

}

}

if (isdigit(str[len - 1]))

{

if (str[len - 1] == '7')

{

seven++;

}

num[n] = str[len - 1];

if (seven < 3)

{

exist = true;

cout << num;

}

}

if (!exist)

{

cout << "Отсутствуют" << endl;

}

cout << endl;

}

void clear(char word[], int len)

{

for (int i = 0; i < len + 1; i++)

{

word[i] = {};

}

}

**Тест:**

Введите строку:

47bdk5767ker8757775lae77

Числа, содержащие не более двух 7:

47 5767 77

**Задача 2.2**

Написать и протестировать аналог функции STRCSPN().

**Блок-схема:**





**Листинг:**

#include <iostream>

using namespace std;

const int MAXLEN = 255;

int mystrcspn(char str[], char keys[]);

void main()

{

setlocale(LC\_ALL, "Russian");

cout << "Введите строку: ";

char str[MAXLEN]{};

cin.getline(str, MAXLEN, '\n');

cout << "Введите строку с символами, которые нужно найти: ";

char keys[MAXLEN]{};

cin.getline(keys, MAXLEN, '\n');

cout << "Символов до первого вхождения: "

<< mystrcspn(str, keys) << endl;

}

int mystrcspn(char str[], char keys[])

{

int slen = strlen(str);

int klen = strlen(keys);

bool find = false;

int len = 0;

while (!find)

{

for (int j = 0; j < klen + 1; j++)

{

if (str[len] == keys[j])

{

find = true;

}

}

len++;

}

len--;

return len;

}

**Тест:**

**#1**

Введите строку: abcb

Введите строку с символами, которые нужно найти: cb

Символов до первого вхождения: 1

**#2**

Введите строку: abcde

Введите строку с символами, которые нужно найти: klmn

Символов до первого вхождения: 5

**Задача 2.3**

Дан произвольный текст. Напечатать в алфавитном порядке все буквы, которые входят в этот текст по одному разу.

**Блок-схема:**









**Листинг:**

#include <iostream>

using namespace std;

const int MAXLEN = 255;

const int ALPHALEN = 26;

bool findsymbol(char str[], char sym);

void sort(char str[]);

void myswap(char& a, char& b);

void main()

{

setlocale(LC\_ALL, "Russian");

cout << "Введите текст: " << endl;

char text[MAXLEN]{};

cin.getline(text, MAXLEN, '\n');

\_strlwr\_s(text);

char Letters[ALPHALEN]{};

int l = 0;

char BlackList[ALPHALEN]{};

int b = 0;

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

{

if (isalpha(text[i]))

{

if (findsymbol(Letters, text[i]) &&

!findsymbol(BlackList, text[i]))

{

BlackList[b] = text[i];

b++;

}

else if (!findsymbol(Letters, text[i]))

{

Letters[l] = text[i];

l++;

}

}

}

sort(Letters);

cout << "Буквы, встречающиеся 1 раз: " << endl;

for (int j = 0; j < strlen(Letters); j++)

{

if (!findsymbol(BlackList, Letters[j]))

{

cout << Letters[j] << " ";

}

}

cout << endl;

}

bool findsymbol(char str[], char sym)

{

bool find = false;

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

{

if (str[i] == sym)

{

find = true;

}

}

return find;

}

void sort(char str[])

{

bool check = true;

int start = 0;

int end = strlen(str) - 1;

while (check)

{

check = false;

for (int p = start; p < end; p++)

{

if (str[p] > str[p + 1])

{

check = true;

myswap(str[p], str[p + 1]);

}

}

start++;

for (int q = end; q > start - 1; q--)

{

if (str[q] < str[q - 1])

{

check = true;

myswap(str[q], str[q - 1]);

}

}

end--;

}

}

void myswap(char& a, char& b)

{

char temp = a;

a = b;

b = temp;

}

**Тест:**

**#1**

Введите текст:

I am gamer

Буквы, встречающиеся 1 раз:

e g i r

**#2**

Введите текст:

Ann not gotta go

Буквы, встречающиеся 1 раз:

**Задача 2.4**

Распечатать те слова, в которых либо буквы упорядочены по алфавиту, либо каждая буква входит в слово не менее двух раз (т.е. слова типа BEER, ABBA).

**Блок-схема:**















**Листинг:**

#include <iostream>

using namespace std;

const int MAXLEN = 255;

const int ALPHALEN = 26;

void clear(char word[]);

bool findsymbol(char str[], char sym);

void main()

{

setlocale(LC\_ALL, "Russian");

cout << "Введите текст: " << endl;

char Text[MAXLEN]{};

cin.getline(Text, MAXLEN, '\n');

int len = strlen(Text);

char Word[MAXLEN]{};

int w = 0;

char Letters[MAXLEN]{};

int l = 0;

bool sorted = true;

char WhiteList[ALPHALEN]{};

int wl = 0;

cout << "Слова, в которых буквы упорядочены по алфавиту"

<<endl << "или каждая буква встречается больше раза: "

<< endl;

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

{

if (isalnum(Text[i]))

{

Word[w] = Text[i];

if (isalpha(Word[w]))

{

if (findsymbol(Letters, tolower(Word[w]))

&& !findsymbol(WhiteList,

tolower(Word[w])))

{

WhiteList[wl] = tolower(Word[w]);

wl++;

}

Letters[l] = tolower(Word[w]);

if (l != 0)

{

if (Letters[l - 1] > Letters[l])

{

sorted = false;

}

}

l++;

}

w++;

if (!isalnum(Text[i + 1]) && strlen(Word) != 0)

{

bool allmorethan1 = true;

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

{

if (!findsymbol(WhiteList, Letters[i]))

{

allmorethan1 = false;

}

}

if (sorted || allmorethan1)

{

cout << Word << " ";

}

sorted = true;

clear(Word);

w = 0;

clear(Letters);

l = 0;

clear(WhiteList);

wl = 0;

}

}

}

if (isalnum(Text[len - 1]))

{

Word[w] = Text[len - 1];

if (isalpha(Word[w]))

{

if (findsymbol(Letters, tolower(Word[w])) &&

!findsymbol(WhiteList, tolower(Word[w])))

{

WhiteList[wl] = tolower(Word[w]);

}

Letters[l] = tolower(Word[w]);

if (l != 0)

{

if (Letters[l - 1] > Letters[l])

{

sorted = false;

}

}

}

bool allmorethan1 = true;

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

{

if (!findsymbol(WhiteList, Letters[i]))

{

allmorethan1 = false;

}

}

if (sorted || allmorethan1)

{

cout << Word << " ";

}

}

cout << endl;

}

void clear(char word[])

{

int len = strlen(word);

for (int i = 0; i < len + 1; i++)

{

word[i] = {};

}

}

bool findsymbol(char str[], char sym)

{

bool find = false;

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

{

if (str[i] == sym)

{

find = true;

}

}

return find;

}

**Тест:**

Введите текст:

ant yoyo code Ann

Слова, в которых буквы упорядочены по алфавиту

или каждая буква встречается больше раза:

ant yoyo Ann

**Вывод:** были получены практические навыки создания алгоритмов сортировки, создания и использования собственных функций, обработки двумерных массивов и строк.