Todo list

Программирование

М.В.Булгакова

18 ноября 2015 г.

Глава 1

Основные конструкции языка

1.1 Задание 1

1.1.1 Задание

Пользователь задает три корня кубического уравнения $x^3 + bx^2 + cx + d$ (например, 1, 2, 3). Вывести значения b, c и d, например: b=-6, c=11, d=-6.

1.1.2 Теоритические сведения

При выполнинии задания в main.c использовался switch для предаставления выбора пользователю вида теста программы(ручной ввод или автоматический). В

`poisk_znacheniy.c`

использовались операторы условного перехода if и циклы for для нахождения b, c и d.

1.1.3 Проектирование

В main.c у пользователя запрашивают режим работы программы, состоящий из:

1. Ручной ввод значений
2. Автоматические тесты

В

`poisk_znacheniy.c`

реализовано взаимодейтсвие с пользователем, считывая введенные значения с консоли,

`poisk_znacheniy.c`

производит поиск коэффициентов уравнений. Модульные тесты находятся в test.c.

1.1.4 Описание тестового стенда и методики тестирования

Для создания проекта использовались Qt Creator 3.5.0 (opensource) и GCC. Пользователь может выбрать один из режимов работы программы

1. Ручной ввод значений
2. Автоматические тесты

Автоматические тесты, на подобие модульных, контролируют исправность программы.

1.1.5 Тестовый план и результаты тестирования

При вызове автоматического теста программа обращается к процедуре

```
void automate_test_variant7_1()
```

. Данная процедура вызывает процедуру

```
void test_poisk_variant7_1()
```

, в которой по уже заданным значениям производится поиск коэффициентов и сравнение рузельтатов с помощью процедуры

```
void test_result_variant7_1(int expected, int actual)
```

, которая выводит на экран "Ok если полученное значение совпало с ожидаемым, в противном случае выводит "Test fail".

1.1.6 Выводы

При написании данной работы были приобретены навыки работы с отладкой(debug), навыки создания модульных тестов и умение разбивать задачи на подзадачи, отделяя общение с пользователем от бизнес-логики.

Листинги

```c
#include <stdio.h>
#include "strange_function.h"
#include "test.h"
#include "poisk_znacheniy.h"
#include "poisk_ugrozi.h"
#include "max_vozmojnoe.h"
#include "zamena_elemetov_mass.h"
#include "main_menu.h"

int main(int argc, char* argv[])
{
    printf(" \n argc =  %d \n", argc);

    int i;
    for(i =0; i< argc; i++)
        printf("\n %d value is  %s \n", i,  argv[i]);

    if(argc == 2){

        if(strcmp(argv[1], "--interactive") == 0){
            main_menu();
        }
    }

    if(strcmp(argv[1], "--is-factorial") == 0){
        fact();
        return(0);
    }

    if(strcmp(argv[1], "--is-strange_function") == 0){
        strange();
        return(0);
    }

    if(strcmp(argv[1], "--is-max_vozmojnoe") == 0){
        max_vozmojnoe();
        return(0);
    }

    if(strcmp(argv[1], "--is-poisk_ugrozi") == 0){
        poisk_ugrozi();
        return(0);
    }
    if(strcmp(argv[1], "--is-zamena_elementov_mass") == 0){
        zamena_elementov_mass();
        return(0);
    }
```

```
48
49
50
51 return 0;
52 }
```

```
1  int poisk_i(int x1, int x2, int x3)
2  {
3      int result1, result2, result3;
4      int j;
5      int i, i_rez;
6      int k;
7      for(i=-100; i <100 ; i++) {
8          for(j=-100; j <100 ; j++) {
9              for(k=-100; k <100 ; k++) {
10                 result1 = x1*x1*x1 + x1*x1*i+x1*j + k;
11                 result2 = x2*x2*x2 + x2*x2*i+x2*j + k;
12                 result3 = x3*x3*x3 + x3*x3*i+x3*j + k;
13                 if (result1 == 0) {
14                     if(result2 == 0){
15                         if (result3 == 0 )
16                         {
17                             i_rez = i;
18                         }
19                     }
20                 }
21             }
22         }
23     }
24     return i_rez;
25 }
26
27 int poisk_j(int x1, int x2, int x3)
28 {
29     int result1, result2, result3;
30     int j, j_rez;
31     int i;
32     int k;
33     for(i=-100; i <100 ; i++) {
34         for(j=-100; j <100 ; j++) {
35             for(k=-100; k <100 ; k++) {
36                 result1 = x1*x1*x1 + x1*x1*i+x1*j + k;
37                 result2 = x2*x2*x2 + x2*x2*i+x2*j + k;
38                 result3 = x3*x3*x3 + x3*x3*i+x3*j + k;
39                 if (result1 == 0){
40                     if (result2 == 0){
41                         if (result3 == 0 )
42                         {
43                             j_rez = j;
```

```
                        }
                    }
                }
            }
        }
    }
    return j_rez;
}

int poisk_k(int x1, int x2, int x3)
{
    int result1, result2, result3;
    int j;
    int i;
    int k, k_rez;
    for(i=-100; i <100 ; i++) {
        for(j=-100; j <100 ; j++) {
            for(k=-100; k <100 ; k++) {
                result1 = x1*x1*x1 + x1*x1*i+x1*j + k;
                result2 = x2*x2*x2 + x2*x2*i+x2*j + k;
                result3 = x3*x3*x3 + x3*x3*i+x3*j + k;
                if (result1 == 0){
                    if (result2 == 0){
                        if (result3 == 0 )
                        {
                            k_rez = k;
                        }
                    }
                }
            }
        }
    }
    return k_rez;
}

void poisk(){
    puts("Введите 3 значени x через Enter");
    int x1, x2, x3;
    scanf("%d", &x1);
    scanf("%d", &x2);
    scanf("%d", &x3);
    int i;
    i = poisk_i(x1,x2,x3);
    int j;
    j = poisk_j(x1,x2,x3);
    int k;
    k = poisk_k(x1,x2,x3);
    printf("%d \n", i);
    printf("%d \n", j);
```

```
93      printf("%d \n", k);
94  }
```

```
1  #ifndef POISK_ZNACHENIY_H
2  #define POISK_ZNACHENIY_H
3
4  int poisk_i(int, int, int);
5  int poisk_j(int, int, int);
6  int poisk_k(int, int, int);
7  void poisk();
8
9  #endif // POISK_ZNACHENIY_H
```

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "strange_function.h"
4  #include "test.h"
5  #include "poisk_znacheniy.h"
6  #include "poisk_ugrozi.h"
7
8  void automate_test(){
9      test_strange_function();
10     test_fact_function();
11 }
12
13 void test_fact_function(){
14     puts("Автоматический тест для вычисления факториала");
15     int number = 4;
16     int result = fact_function(number);
17     printf("Число: %d, Факториал: %d \n",number,result);
18     test_result(5040, result);
19 }
20
21 void test_strange_function(){
22     int first_number = 20;
23     int second_number = 12;
24     int result = strange_function(first_number, second_number
           );
25     printf("Первое число: %d, Второе число : %d , Результат:
           %d \n", first_number, second_number, result);
26     test_result(8, result);
27     first_number = 7;
28     second_number = 30;
29     result = strange_function(first_number, second_number);
30     printf("Первое число: %d, Второе число : %d , Результат:
           %d \n", first_number, second_number, result);
31     test_result(37, result);
32 }
33
```

```
34 void test_result(int expected, int actual){
35     if (expected == actual)
36         puts("Ok");
37     else
38         puts("Test fail");
39 }
40
41 void automate_test_variant7_1(){
42     test_poisk_variant7_1();
43 }
44
45 void test_poisk_variant7_1(){
46     int x1 = 2;
47     int x2 = 2;
48     int x3 = 2;
49     int i;
50     i= poisk_i(x1,x2,x3);
51     int j;
52     j= poisk_j(x1,x2,x3);
53     int k;
54     k= poisk_k(x1,x2,x3);
55     printf("x1: %d, x2 : %d, x3 : %d, i : %d, j : %d,k: %d \n
           ", x1, x2, x3, i, j, k);
56     test_result_variant7_1(73, i);
57     test_result_variant7_1(-100, j);
58     test_result_variant7_1(-100, k);
59
60 }
61
62 void test_result_variant7_1(int expected, int actual){
63     if (expected == actual)
64         puts("Ok");
65     else
66         puts("Test fail");
67 }
68 void poisk_variant7_1(){
69     puts("Введите 3 числа");
70     int x1, x2, x3;
71     scanf("%d", &x1);
72     scanf("%d", &x2);
73     scanf("%d", &x3);
74     int i;
75     i = poisk_i(x1,x2,x3);
76     int j;
77     j = poisk_j(x1,x2,x3);
78     int k;
79     k = poisk_k(x1,x2,x3);
80     printf("%d \n", i);
81     printf("%d \n", j);
```

```
82      printf("%d \n", k);
83  }
84
85  void automate_test_variant7_2(){
86      test_poisk_variant7_2();
87  }
88
89  void test_poisk_variant7_2(){
90      int x1 = 1;
91      int x2 = 2;
92      int y1 = 3;
93      int y2 = 3;
94      int z1 = 1;
95      int z2 = 3;
96      int a1 = 4;
97      int a2 = 3;
98      int result;
99      if (x1==y1)
100         result=1;
101     if (x1==z1)
102         result=2;
103     if (x1==a1)
104         result=3;
105     if (x2==y2)
106         result=1;
107     if (x2==z2)
108         result=2;
109     if (x2==a2)
110         result=3;
111     printf("x1: %d, x2 : %d, Nomer ladii : %d \n", x1, x2,
            result);
112     test_result_variant7_2(2, result);
113
114 }
115
116 void test_result_variant7_2(int expected, int actual){
117     if (expected == actual)
118         puts("Ok");
119     else
120         puts("Test fail");
121 }
122 void automate_test_max_vozmojnoe(){
123     test_poisk_max_vozmojnoe();
124 }
125
126 void test_poisk_max_vozmojnoe(){
127     int max=0, N=4157, M=8024, N_ostatok_ot_del,
            N_zhelaya_chast, M_ostatok_ot_del, M_zhelaya_chast ;
128     int i=0;
```

```
129     printf("N : %d, M : %d ", N, M);
130     while (N>0) {
131        N_ostatok_ot_del= floor(fmod(N, 10));
132        M_ostatok_ot_del= floor(fmod(M, 10));
133        N_zhelaya_chast=floor(N/10);
134        M_zhelaya_chast=floor(M/10);
135        if(N_ostatok_ot_del>M_ostatok_ot_del)
136            max=max+pow(10,i)*N_ostatok_ot_del;
137        else
138            max=max+pow(10,i)*M_ostatok_ot_del;
139        i=i+1;
140        N=N_zhelaya_chast;
141        M=M_zhelaya_chast;
142     }
143     printf("max : %d \n", max);
144     test_result_max_vozmojnoe(8157, max);
145 
146 }
147 
148 void test_result_max_vozmojnoe(int expected, int actual){
149     if (expected == actual)
150         puts("Ok");
151     else
152         puts("Test fail");
153 }
154 
155 void automate_test_zamena_elementov_mass(){
156    test_poisk_zamena_elementov_mass();
157 }
158 
159 void test_poisk_zamena_elementov_mass(){
160 
161    FILE *mf;
162     mf=fopen("zamena.txt","r");
163     int n, i=0;
164     float *p;
165    fscanf(mf,"%d \n",&n);
166    p = (float *) malloc(n*sizeof(float));
167 
168    for (i = 0; i<=(n-1); i++){
169        fscanf(mf,"%f\n",&p[i]);
170     }
171 
172     fclose(mf);
173 
174     p[0]=(p[0]+p[1])/2;
175     p[n-1]=(p[n-2]+p[n-1])/2;
176 
177     for (i = 1; i<(n-1); i++){
```

```
178       p[i]=(p[i-1]+2*p[i]+p[i+1])/4;
179      }
180     for (i = 0; i<=(n-1); i++){
181     test_result_zamena_elementov_mass(p[i], p[i]);
182     }
183
184 }
185
186 void test_result_zamena_elementov_mass(int expected, int
        actual){
187      if (expected == actual)
188          puts("Ok");
189      else
190          puts("Test fail");
191 }
```

```
1  #ifndef TEST_H
2  #define TEST_H
3
4  void automate_test();
5  void test_fact_function();
6  void test_strange_function();
7  void test_result(int, int);
8  void automate_test_variant7_1();
9  void test_poisk_variant7_1();
10 void test_result_variant7_1(int, int);
11 void poisk_variant7_1();
12 void automate_test_variant7_2();
13 void test_poisk_variant7_2();
14 void test_result_variant7_2(int, int);
15 void automate_test_max_vozmojnoe();
16 void test_poisk_max_vozmojnoe();
17 void test_result_max_vozmojnoe(int, int);
18 void test_poisk_zamena_elementov_mass();
19 void automate_test_zamena_elementov_mass();
20 void test_result_zamena_elementov_mass(int, int);
21
22 #endif // TEST_H
```

1.2 Задание 2

1.2.1 Задание

На шахматной доске стоят черный король и три белые ладьи (ладья бьет по горизонтали и вертикали). Определить, не находится ли король под

боем, а если есть угроза, то от кого именно. Координаты короля и ладей вводить целыми числами.

1.2.2 Теоритические сведения

При выполнинии задания в main.c использовался switch для предаставления выбора пользователю вида теста программы(ручной ввод или автоматический). В

`poisk_ugrozi.c`

использовались операторы условного перехода if для нахождения угрозы королю от ладьи.

1.2.3 Проектирование

В main.c у пользователя запрашивают режим работы программы, состоящий из:

1. Ручной ввод значений
2. Автоматические тесты

В

`poisk_ugrozi.c`

реализовано взаимодейтсвие с пользователем, считывая введенные значения с консоли,

`poisk_ugrozi.c`

производит поиск угрозы королю от одной или нескольких ладей. Модульные тесты находятся в test.c.

1.2.4 Описание тестового стенда и методики тестирования

Для создания проекта использовались Qt Creator 3.5.0 (opensource) и GCC. Пользователь может выбрать один из режимов работы программы

1. Ручной ввод значений
2. Автоматические тесты

Автоматические тесты, на подобие модульных, контролируют исправность программы.

1.2.5 Тестовый план и результаты тестирования

При вызове автоматического теста программа обращается к процедуре

```
void automate_test_variant7_2()
```

. Данная процедура вызывает процедуру

```
void test_poisk_variant7_2()
```

, в которой по уже заданным значениям производится поиск номера ладьи, угрожающей королю, и сравнение рузельтатов с помощью процедуры

```
void test_result_variant7_2(int expected, int actual)
```

, которая выводит на экран "Ok если полученное значение совпало с ожидаемым, в противном случае выводит "Test fail".

1.2.6 Выводы

При написании данной работы были улучшены навыки работы с отладкой(debug), навыки создания модульных тестов и умение разбивать задачи на подзадачи, отделяя общение с пользователем от бизнес-логики.

Листинги

```
#include <stdio.h>
#include "strange_function.h"
#include "test.h"
#include "poisk_znacheniy.h"
#include "poisk_ugrozi.h"
#include "max_vozmojnoe.h"
#include "zamena_elemetov_mass.h"
#include "main_menu.h"

int main(int argc, char* argv[])
{
    printf(" \n argc =  %d \n", argc);

    int i;
    for(i =0; i< argc; i++)
        printf("\n %d value is  %s \n", i,  argv[i]);

    if(argc == 2){

```

```
        if(strcmp(argv[1], "--interactive") == 0){
            main_menu();
        }
    }

    if(strcmp(argv[1], "--is-factorial") == 0){
        fact();
        return(0);
    }

    if(strcmp(argv[1], "--is-strange_function") == 0){
        strange();
        return(0);
    }

    if(strcmp(argv[1], "--is-max_vozmojnoe") == 0){
        max_vozmojnoe();
        return(0);
    }

    if(strcmp(argv[1], "--is-poisk_ugrozi") == 0){
        poisk_ugrozi();
        return(0);
    }
    if(strcmp(argv[1], "--is-zamena_elementov_mass") == 0){
        zamena_elementov_mass();
        return(0);
    }



return 0;
}
```

```
void poisk_ugrozi(){
    puts("Введите 8 цифр, обозначающих позиции короля и ладей
        , через клавишу Enter");
    int x1, x2, y1, y2, z1, z2, a1, a2;
    scanf("%d", &x1);
    scanf("%d", &x2);
    scanf("%d", &y1);
    scanf("%d", &y2);
    scanf("%d", &z1);
    scanf("%d", &z2);
    scanf("%d", &a1);
    scanf("%d", &a2);
    if (x1==y1)
        puts("Угроза от первой ладьи");

```

```
    if (x1==z1)
        puts("Угроза от второй ладьи");
    if (x1==a1)
        puts("Угроза от третьей ладьи");
    if (x2==y2)
        puts("Угроза от первой ладьи");
    if (x2==z2)
        puts("Угроза от второй ладьи");
    if (x2==a2)
        puts("Угроза от третьей ладьи");
}
```

```
#ifndef POISK_UGROZI_H
#define POISK_UGROZI_H

void poisk_ugrozi();

#endif // POISK_UGROZI_H
```

```
#include <stdio.h>
#include <stdlib.h>
#include "strange_function.h"
#include "test.h"
#include "poisk_znacheniy.h"
#include "poisk_ugrozi.h"

void automate_test(){
    test_strange_function();
    test_fact_function();
}

void test_fact_function(){
    puts("Автоматический тест для вычисления факториала");
    int number = 4;
    int result = fact_function(number);
    printf("Число: %d, Факториал: %d \n",number,result);
    test_result(5040, result);
}

void test_strange_function(){
    int first_number = 20;
    int second_number = 12;
    int result = strange_function(first_number, second_number
        );
    printf("Первое число: %d, Второе число : %d , Результат:
        %d \n", first_number, second_number, result);
    test_result(8, result);
    first_number = 7;
    second_number = 30;
```

```
29      result = strange_function(first_number, second_number);
30      printf("Первое число: %d, Второе число : %d , Результат:
            %d \n", first_number, second_number, result);
31      test_result(37, result);
32  }
33
34  void test_result(int expected, int actual){
35      if (expected == actual)
36          puts("Ok");
37      else
38          puts("Test fail");
39  }
40
41  void automate_test_variant7_1(){
42      test_poisk_variant7_1();
43  }
44
45  void test_poisk_variant7_1(){
46      int x1 = 2;
47      int x2 = 2;
48      int x3 = 2;
49      int i;
50      i= poisk_i(x1,x2,x3);
51      int j;
52      j= poisk_j(x1,x2,x3);
53      int k;
54      k= poisk_k(x1,x2,x3);
55      printf("x1: %d, x2 : %d, x3 : %d, i : %d, j : %d,k: %d \n
            ", x1, x2, x3, i, j, k);
56      test_result_variant7_1(73, i);
57      test_result_variant7_1(-100, j);
58      test_result_variant7_1(-100, k);
59
60  }
61
62  void test_result_variant7_1(int expected, int actual){
63      if (expected == actual)
64          puts("Ok");
65      else
66          puts("Test fail");
67  }
68  void poisk_variant7_1(){
69      puts("Введите 3 числа");
70      int x1, x2, x3;
71      scanf("%d", &x1);
72      scanf("%d", &x2);
73      scanf("%d", &x3);
74      int i;
75      i = poisk_i(x1,x2,x3);
```

```
76      int j;
77      j = poisk_j(x1,x2,x3);
78      int k;
79      k = poisk_k(x1,x2,x3);
80      printf("%d \n", i);
81      printf("%d \n", j);
82      printf("%d \n", k);
83  }
84
85  void automate_test_variant7_2(){
86      test_poisk_variant7_2();
87  }
88
89  void test_poisk_variant7_2(){
90      int x1 = 1;
91      int x2 = 2;
92      int y1 = 3;
93      int y2 = 3;
94      int z1 = 1;
95      int z2 = 3;
96      int a1 = 4;
97      int a2 = 3;
98      int result;
99      if (x1==y1)
100         result=1;
101     if (x1==z1)
102         result=2;
103     if (x1==a1)
104         result=3;
105     if (x2==y2)
106         result=1;
107     if (x2==z2)
108         result=2;
109     if (x2==a2)
110         result=3;
111     printf("x1: %d, x2 : %d, Nomer ladii : %d \n", x1, x2,
        result);
112     test_result_variant7_2(2, result);
113
114 }
115
116 void test_result_variant7_2(int expected, int actual){
117     if (expected == actual)
118         puts("Ok");
119     else
120         puts("Test fail");
121 }
122 void automate_test_max_vozmojnoe(){
123     test_poisk_max_vozmojnoe();
```

```
124 }
125
126 void test_poisk_max_vozmojnoe(){
127     int max=0, N=4157, M=8024, N_ostatok_ot_del,
          N_zhelaya_chast, M_ostatok_ot_del, M_zhelaya_chast ;
128     int i=0;
129     printf("N : %d, M : %d ", N, M);
130     while (N>0) {
131        N_ostatok_ot_del= floor(fmod(N, 10));
132        M_ostatok_ot_del= floor(fmod(M, 10));
133        N_zhelaya_chast=floor(N/10);
134        M_zhelaya_chast=floor(M/10);
135        if(N_ostatok_ot_del>M_ostatok_ot_del)
136            max=max+pow(10,i)*N_ostatok_ot_del;
137        else
138            max=max+pow(10,i)*M_ostatok_ot_del;
139        i=i+1;
140        N=N_zhelaya_chast;
141        M=M_zhelaya_chast;
142     }
143     printf("max : %d \n", max);
144     test_result_max_vozmojnoe(8157, max);
145
146 }
147
148 void test_result_max_vozmojnoe(int expected, int actual){
149     if (expected == actual)
150         puts("Ok");
151     else
152         puts("Test fail");
153 }
154
155 void automate_test_zamena_elementov_mass(){
156    test_poisk_zamena_elementov_mass();
157 }
158
159 void test_poisk_zamena_elementov_mass(){
160
161    FILE *mf;
162     mf=fopen("zamena.txt","r");
163     int n, i=0;
164     float *p;
165    fscanf(mf,"%d \n",&n);
166    p = (float *) malloc(n*sizeof(float));
167
168    for (i = 0; i<=(n-1); i++){
169        fscanf(mf,"%f\n",&p[i]);
170     }
171
```

```
172     fclose(mf);
173
174     p[0]=(p[0]+p[1])/2;
175     p[n-1]=(p[n-2]+p[n-1])/2;
176
177     for (i = 1; i<(n-1); i++){
178      p[i]=(p[i-1]+2*p[i]+p[i+1])/4;
179     }
180    for (i = 0; i<=(n-1); i++){
181    test_result_zamena_elementov_mass(p[i], p[i]);
182    }
183
184 }
185
186 void test_result_zamena_elementov_mass(int expected, int
       actual){
187     if (expected == actual)
188         puts("Ok");
189     else
190         puts("Test fail");
191 }
```

```
1  #ifndef TEST_H
2  #define TEST_H
3
4  void automate_test();
5  void test_fact_function();
6  void test_strange_function();
7  void test_result(int, int);
8  void automate_test_variant7_1();
9  void test_poisk_variant7_1();
10 void test_result_variant7_1(int, int);
11 void poisk_variant7_1();
12 void automate_test_variant7_2();
13 void test_poisk_variant7_2();
14 void test_result_variant7_2(int, int);
15 void automate_test_max_vozmojnoe();
16 void test_poisk_max_vozmojnoe();
17 void test_result_max_vozmojnoe(int, int);
18 void test_poisk_zamena_elementov_mass();
19 void automate_test_zamena_elementov_mass();
20 void test_result_zamena_elementov_mass(int, int);
21
22 #endif // TEST_H
```

Глава 2

Циклы

2.1 Задание 1

2.1.1 Задание

Составить из соответствующих цифр чисел M и N наибольшее возможное число. Примеры: 4157, 8024 > 8157; 323, 10714 > 10724.

2.1.2 Теоритические сведения

При выполнинии задания в main.c использовался switch для предаставления выбора пользователю вида теста программы(ручной ввод или автоматический). В

`max_vozmojnoe.c`

использовались операторы условного перехода if и switch, циклы while и математические функции floor, fmod, pow для нахождения наибольшего возможного числа.

2.1.3 Проектирование

В main.c у пользователя запрашивают режим работы программы, состоящий из:

1. Ручной ввод значений
2. Автоматические тесты

В

`max_vozmojnoe.c`

реализовано взаимодейтсвие с пользователем, считывая введенные значения с консоли,

`max_vozmojnoe.c`

производит поиск ниабольшего возможного числа, составленного из чисел M и N. Модульные тесты находятся в test.c.

2.1.4 Описание тестового стенда и методики тестирования

Для создания проекта использовались Qt Creator 3.5.0 (opensource) и GCC. Пользователь может выбрать один из режимов работы программы

1. Ручной ввод значений
2. Автоматические тесты

Автоматические тесты, на подобие модульных, контролируют исправность программы.

2.1.5 Тестовый план и результаты тестирования

При вызове автоматического теста программа обращается к процедуре

```
void automate_test_max_vozmojnoe()
```

. Данная процедура вызывает процедуру

```
void test_poisk_max_vozmojnoe()
```

, в которой по уже заданным значениям производится поиск наибольшего возможного числа, состоящего из M и N, и сравнение рузельтатов с помощью процедуры

```
void test_result_max_vozmojnoe(int expected, int actual)
```

, которая выводит на экран "Ok если полученное значение совпало с ожидаемым, в противном случае выводит "Test fail".

2.1.6 Выводы

При написании данной работы были получены навыки работы со стандартной библиотекой math.h, навыки создания циклов while.

Листинги

```c
#include <stdio.h>
#include "strange_function.h"
#include "test.h"
#include "poisk_znacheniy.h"
#include "poisk_ugrozi.h"
#include "max_vozmojnoe.h"
#include "zamena_elemetov_mass.h"
#include "main_menu.h"

int main(int argc, char* argv[])
{
    printf(" \n argc =  %d \n", argc);

    int i;
    for(i =0; i< argc; i++)
        printf("\n %d value is  %s \n", i,  argv[i]);

    if(argc == 2){

        if(strcmp(argv[1], "--interactive") == 0){
            main_menu();
        }
    }

    if(strcmp(argv[1], "--is-factorial") == 0){
        fact();
        return(0);
    }

    if(strcmp(argv[1], "--is-strange_function") == 0){
        strange();
        return(0);
    }

    if(strcmp(argv[1], "--is-max_vozmojnoe") == 0){
        max_vozmojnoe();
        return(0);
    }

    if(strcmp(argv[1], "--is-poisk_ugrozi") == 0){
        poisk_ugrozi();
        return(0);
    }
    if(strcmp(argv[1], "--is-zamena_elementov_mass") == 0){
        zamena_elementov_mass();
        return(0);
    }
```

```
48
49
50
51 return 0;
52 }
```

```
1  #include <math.h>
2  void max_vozmojnoe(){
3      puts("Введите числа M и N через Enter");
4      int max=0, N, M, N_ostatok_ot_del, N_zhelaya_chast,
           M_ostatok_ot_del, M_zhelaya_chast ;
5      int i=0,k;
6      scanf("%d", &N);
7      scanf("%d", &M);
8      if (M>N) k=1; else k=0;
9      switch (k) {
10     case 1: while (N>0) {
11             N_ostatok_ot_del= floor(fmod(N, 10));
12             M_ostatok_ot_del= floor(fmod(M, 10));
13             N_zhelaya_chast=floor(N/10);
14             M_zhelaya_chast=floor(M/10);
15             if(N_ostatok_ot_del>M_ostatok_ot_del)
16                 max=max+pow(10,i)*N_ostatok_ot_del;
17             else
18                 max=max+pow(10,i)*M_ostatok_ot_del;
19             i=i+1;
20             N=N_zhelaya_chast;
21             M=M_zhelaya_chast;
22         }
23         max=max+pow(10,i)*M;
24         printf("Максимальное число = %d",max);
25         break;
26     case 0: while (M>0) {
27             N_ostatok_ot_del= floor(fmod(N, 10));
28             M_ostatok_ot_del= floor(fmod(M, 10));
29             N_zhelaya_chast=floor(N/10);
30             M_zhelaya_chast=floor(M/10);
31             if(N_ostatok_ot_del>M_ostatok_ot_del)
32                 max=max+pow(10,i)*N_ostatok_ot_del;
33             else
34                 max=max+pow(10,i)*M_ostatok_ot_del;
35             i=i+1;
36             N=N_zhelaya_chast;
37             M=M_zhelaya_chast;
38         }
39         max=max+pow(10,i)*N;
40         printf("Максимальное число = %d",max);
41         break;
42     }
```

```
}
```

```
#ifndef MAX_VOZMOJNOE_H
#define MAX_VOZMOJNOE_H
void max_vozmojnoe();
#endif // MAX_VOZMOJNOE_H
```

```
#include <stdio.h>
#include <stdlib.h>
#include "strange_function.h"
#include "test.h"
#include "poisk_znacheniy.h"
#include "poisk_ugrozi.h"

void automate_test(){
    test_strange_function();
    test_fact_function();
}

void test_fact_function(){
    puts("Автоматический тест для вычисления факториала");
    int number = 4;
    int result = fact_function(number);
    printf("Число: %d, Факториал: %d \n",number,result);
    test_result(5040, result);
}

void test_strange_function(){
    int first_number = 20;
    int second_number = 12;
    int result = strange_function(first_number, second_number
        );
    printf("Первое число: %d, Второе число : %d , Результат:
        %d \n", first_number, second_number, result);
    test_result(8, result);
    first_number = 7;
    second_number = 30;
    result = strange_function(first_number, second_number);
    printf("Первое число: %d, Второе число : %d , Результат:
        %d \n", first_number, second_number, result);
    test_result(37, result);
}

void test_result(int expected, int actual){
    if (expected == actual)
        puts("Ok");
    else
        puts("Test fail");
}
```

```
40
41 void automate_test_variant7_1(){
42     test_poisk_variant7_1();
43 }
44
45 void test_poisk_variant7_1(){
46     int x1 = 2;
47     int x2 = 2;
48     int x3 = 2;
49     int i;
50     i= poisk_i(x1,x2,x3);
51     int j;
52     j= poisk_j(x1,x2,x3);
53     int k;
54     k= poisk_k(x1,x2,x3);
55     printf("x1: %d, x2 : %d, x3 : %d, i : %d, j : %d,k: %d \n
           ", x1, x2, x3, i, j, k);
56     test_result_variant7_1(73, i);
57     test_result_variant7_1(-100, j);
58     test_result_variant7_1(-100, k);
59
60 }
61
62 void test_result_variant7_1(int expected, int actual){
63     if (expected == actual)
64         puts("Ok");
65     else
66         puts("Test fail");
67 }
68 void poisk_variant7_1(){
69     puts("Введите 3 числа");
70     int x1, x2, x3;
71     scanf("%d", &x1);
72     scanf("%d", &x2);
73     scanf("%d", &x3);
74     int i;
75     i = poisk_i(x1,x2,x3);
76     int j;
77     j = poisk_j(x1,x2,x3);
78     int k;
79     k = poisk_k(x1,x2,x3);
80     printf("%d \n", i);
81     printf("%d \n", j);
82     printf("%d \n", k);
83 }
84
85 void automate_test_variant7_2(){
86     test_poisk_variant7_2();
87 }
```

```
88
89 void test_poisk_variant7_2(){
90     int x1 = 1;
91     int x2 = 2;
92     int y1 = 3;
93     int y2 = 3;
94     int z1 = 1;
95     int z2 = 3;
96     int a1 = 4;
97     int a2 = 3;
98     int result;
99     if (x1==y1)
100         result=1;
101     if (x1==z1)
102         result=2;
103     if (x1==a1)
104         result=3;
105     if (x2==y2)
106         result=1;
107     if (x2==z2)
108         result=2;
109     if (x2==a2)
110         result=3;
111     printf("x1: %d, x2 : %d, Nomer ladii : %d \n", x1, x2,
        result);
112     test_result_variant7_2(2, result);
113
114 }
115
116 void test_result_variant7_2(int expected, int actual){
117     if (expected == actual)
118         puts("Ok");
119     else
120         puts("Test fail");
121 }
122 void automate_test_max_vozmojnoe(){
123     test_poisk_max_vozmojnoe();
124 }
125
126 void test_poisk_max_vozmojnoe(){
127     int max=0, N=4157, M=8024, N_ostatok_ot_del,
        N_zhelaya_chast, M_ostatok_ot_del, M_zhelaya_chast ;
128     int i=0;
129     printf("N : %d, M : %d ", N, M);
130     while (N>0) {
131        N_ostatok_ot_del= floor(fmod(N, 10));
132        M_ostatok_ot_del= floor(fmod(M, 10));
133        N_zhelaya_chast=floor(N/10);
134        M_zhelaya_chast=floor(M/10);
```

```
135         if(N_ostatok_ot_del>M_ostatok_ot_del)
136             max=max+pow(10,i)*N_ostatok_ot_del;
137         else
138             max=max+pow(10,i)*M_ostatok_ot_del;
139         i=i+1;
140         N=N_zhelaya_chast;
141         M=M_zhelaya_chast;
142     }
143     printf("max : %d \n", max);
144     test_result_max_vozmojnoe(8157, max);
145 
146 }
147 
148 void test_result_max_vozmojnoe(int expected, int actual){
149     if (expected == actual)
150         puts("Ok");
151     else
152         puts("Test fail");
153 }
154 
155 void automate_test_zamena_elementov_mass(){
156    test_poisk_zamena_elementov_mass();
157 }
158 
159 void test_poisk_zamena_elementov_mass(){
160 
161    FILE *mf;
162     mf=fopen("zamena.txt","r");
163     int n, i=0;
164     float *p;
165    fscanf(mf,"%d \n",&n);
166    p = (float *) malloc(n*sizeof(float));
167 
168    for (i = 0; i<=(n-1); i++){
169        fscanf(mf,"%f\n",&p[i]);
170     }
171 
172     fclose(mf);
173 
174     p[0]=(p[0]+p[1])/2;
175     p[n-1]=(p[n-2]+p[n-1])/2;
176 
177     for (i = 1; i<(n-1); i++){
178      p[i]=(p[i-1]+2*p[i]+p[i+1])/4;
179     }
180    for (i = 0; i<=(n-1); i++){
181    test_result_zamena_elementov_mass(p[i], p[i]);
182    }
183 
```

```
184 }
185
186 void test_result_zamena_elementov_mass(int expected, int
    actual){
187     if (expected == actual)
188         puts("Ok");
189     else
190         puts("Test fail");
191 }
```

```
 1 #ifndef TEST_H
 2 #define TEST_H
 3
 4 void automate_test();
 5 void test_fact_function();
 6 void test_strange_function();
 7 void test_result(int, int);
 8 void automate_test_variant7_1();
 9 void test_poisk_variant7_1();
10 void test_result_variant7_1(int, int);
11 void poisk_variant7_1();
12 void automate_test_variant7_2();
13 void test_poisk_variant7_2();
14 void test_result_variant7_2(int, int);
15 void automate_test_max_vozmojnoe();
16 void test_poisk_max_vozmojnoe();
17 void test_result_max_vozmojnoe(int, int);
18 void test_poisk_zamena_elementov_mass();
19 void automate_test_zamena_elementov_mass();
20 void test_result_zamena_elementov_mass(int, int);
21
22 #endif // TEST_H
```

Глава 3

Массивы

3.1 Задание 1

3.1.1 Задание

Каждый элемент вектора A(n) (кроме двух крайних) заменить выражением: $a_i = (a_(i-1)+2a_i+a_(i+1))/4$, а крайние элементы – выражениями: $a_1 = (a_1 + a_2)/2, a_n = (a_(n-1) + a_n)/2$.

3.1.2 Теоритические сведения

При выполнинии задания в main.c использовался switch для предаставления выбора пользователю вида теста программы(ручной ввод или автоматический). В

`zamena_elementov_mass.c`

использовались циклы for и функции работы с файлами fscanf, fopen, fclose и процедуры задания и освобождения памяти malloc и free.

3.1.3 Проектирование

В main.c у пользователя запрашивают режим работы программы, состоящий из:

1. Ручной ввод значений
2. Автоматические тесты

В

`zamena_elementov_mass.c`

реализовано взаимодейтсвие с пользовательским файлом, считывая введенные значения с файла zamena.txt,

`zamena_elementov_mass.c`

производит замену элементов по заданному условию. Модульные тесты находятся в test.c.

3.1.4 Описание тестового стенда и методики тестирования

Для создания проекта использовались Qt Creator 3.5.0 (opensource) и GCC. Пользователь может выбрать один из режимов работы программы

1. Ручной ввод значений
2. Автоматические тесты

Автоматические тесты, на подобие модульных, контролируют исправность программы.

3.1.5 Тестовый план и результаты тестирования

При вызове автоматического теста программа обращается к процедуре

```
void automate_test_zamena_elementov_mass()
```

. Данная процедура вызывает процедуру

```
void test_poisk_zamena_elementov_mass()
```

, в которой по уже заданным в файле значениям производит замену значений элементов массива, и сравнение рузельтатов с помощью процедуры

```
void test_result_zamena_elementov_mass()(int expected, int actual)
```

, которая выводит на экран "Ok если полученное значение совпало с ожидаемым, в противном случае выводит "Test fail".

3.1.6 Выводы

При написании данной работы были получены навыки работы с файлами и массивами.

Листинги

```c
#include <stdio.h>
#include "strange_function.h"
#include "test.h"
#include "poisk_znacheniy.h"
#include "poisk_ugrozi.h"
#include "max_vozmojnoe.h"
#include "zamena_elemetov_mass.h"
#include "main_menu.h"

int main(int argc, char* argv[])
{
    printf(" \n argc =  %d \n", argc);

    int i;
    for(i =0; i< argc; i++)
        printf("\n %d value is  %s \n", i,  argv[i]);

    if(argc == 2){

        if(strcmp(argv[1], "--interactive") == 0){
            main_menu();
        }
    }

    if(strcmp(argv[1], "--is-factorial") == 0){
        fact();
        return(0);
    }

    if(strcmp(argv[1], "--is-strange_function") == 0){
        strange();
        return(0);
    }

    if(strcmp(argv[1], "--is-max_vozmojnoe") == 0){
        max_vozmojnoe();
        return(0);
    }

    if(strcmp(argv[1], "--is-poisk_ugrozi") == 0){
        poisk_ugrozi();
        return(0);
    }
    if(strcmp(argv[1], "--is-zamena_elementov_mass") == 0){
        zamena_elementov_mass();
        return(0);
    }
```

```
48
49
50
51 return 0;
52 }
```

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  void zamena_elementov_mass(){
4     FILE *mf;
5     mf=fopen("zamena.txt","r");
6     int n, i=0;
7     float *p;
8     fscanf(mf,"%d \n",&n);
9     p = (float *) malloc(n*sizeof(float));
10
11    for (i = 0; i<=(n-1); i++){
12        fscanf(mf,"%f\n",&p[i]);
13    }
14
15    fclose(mf);
16
17    p[0]=(p[0]+p[1])/2;
18    p[n-1]=(p[n-2]+p[n-1])/2;
19
20    for (i = 1; i<(n-1); i++){
21       p[i]=(p[i-1]+2*p[i]+p[i+1])/4;
22    }
23
24    for (i = 0; i<=(n-1); i++){
25        printf("%f\n",*(p+i));
26    }
27
28   free(p);
29 }
```

```
1 #ifndef ZAMENA_ELEMETOV_MASS_H
2 #define ZAMENA_ELEMETOV_MASS_H
3 void zamena_elementov_mass();
4 #endif // ZAMENA_ELEMETOV_MASS_H
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "strange_function.h"
4 #include "test.h"
5 #include "poisk_znacheniy.h"
6 #include "poisk_ugrozi.h"
7
```

```
 8 void automate_test(){
 9     test_strange_function();
10     test_fact_function();
11 }
12
13 void test_fact_function(){
14     puts("Автоматический тест для вычисления факториала");
15     int number = 4;
16     int result = fact_function(number);
17     printf("Число: %d, Факториал: %d \n",number,result);
18     test_result(5040, result);
19 }
20
21 void test_strange_function(){
22     int first_number = 20;
23     int second_number = 12;
24     int result = strange_function(first_number, second_number
           );
25     printf("Первое число: %d, Второе число : %d , Результат:
           %d \n", first_number, second_number, result);
26     test_result(8, result);
27     first_number = 7;
28     second_number = 30;
29     result = strange_function(first_number, second_number);
30     printf("Первое число: %d, Второе число : %d , Результат:
           %d \n", first_number, second_number, result);
31     test_result(37, result);
32 }
33
34 void test_result(int expected, int actual){
35     if (expected == actual)
36         puts("Ok");
37     else
38         puts("Test fail");
39 }
40
41 void automate_test_variant7_1(){
42     test_poisk_variant7_1();
43 }
44
45 void test_poisk_variant7_1(){
46     int x1 = 2;
47     int x2 = 2;
48     int x3 = 2;
49     int i;
50     i= poisk_i(x1,x2,x3);
51     int j;
52     j= poisk_j(x1,x2,x3);
53     int k;
```

```
54      k= poisk_k(x1,x2,x3);
55      printf("x1: %d, x2 : %d, x3 : %d, i : %d, j : %d,k: %d \n
            ", x1, x2, x3, i, j, k);
56      test_result_variant7_1(73, i);
57      test_result_variant7_1(-100, j);
58      test_result_variant7_1(-100, k);
59
60  }
61
62  void test_result_variant7_1(int expected, int actual){
63      if (expected == actual)
64          puts("Ok");
65      else
66          puts("Test fail");
67  }
68  void poisk_variant7_1(){
69      puts("Введите 3 числа");
70      int x1, x2, x3;
71      scanf("%d", &x1);
72      scanf("%d", &x2);
73      scanf("%d", &x3);
74      int i;
75      i = poisk_i(x1,x2,x3);
76      int j;
77      j = poisk_j(x1,x2,x3);
78      int k;
79      k = poisk_k(x1,x2,x3);
80      printf("%d \n", i);
81      printf("%d \n", j);
82      printf("%d \n", k);
83  }
84
85  void automate_test_variant7_2(){
86      test_poisk_variant7_2();
87  }
88
89  void test_poisk_variant7_2(){
90      int x1 = 1;
91      int x2 = 2;
92      int y1 = 3;
93      int y2 = 3;
94      int z1 = 1;
95      int z2 = 3;
96      int a1 = 4;
97      int a2 = 3;
98      int result;
99      if (x1==y1)
100         result=1;
101     if (x1==z1)
```

```
102             result=2;
103         if (x1==a1)
104             result=3;
105         if (x2==y2)
106             result=1;
107         if (x2==z2)
108             result=2;
109         if (x2==a2)
110             result=3;
111         printf("x1: %d, x2 : %d, Nomer ladii : %d \n", x1, x2,
                result);
112         test_result_variant7_2(2, result);
113 
114 }
115 
116 void test_result_variant7_2(int expected, int actual){
117         if (expected == actual)
118             puts("Ok");
119         else
120             puts("Test fail");
121 }
122 void automate_test_max_vozmojnoe(){
123         test_poisk_max_vozmojnoe();
124 }
125 
126 void test_poisk_max_vozmojnoe(){
127         int max=0, N=4157, M=8024, N_ostatok_ot_del,
                N_zhelaya_chast, M_ostatok_ot_del, M_zhelaya_chast ;
128         int i=0;
129         printf("N : %d, M : %d ", N, M);
130         while (N>0) {
131             N_ostatok_ot_del= floor(fmod(N, 10));
132             M_ostatok_ot_del= floor(fmod(M, 10));
133             N_zhelaya_chast=floor(N/10);
134             M_zhelaya_chast=floor(M/10);
135             if(N_ostatok_ot_del>M_ostatok_ot_del)
136                 max=max+pow(10,i)*N_ostatok_ot_del;
137             else
138                 max=max+pow(10,i)*M_ostatok_ot_del;
139             i=i+1;
140             N=N_zhelaya_chast;
141             M=M_zhelaya_chast;
142         }
143         printf("max : %d \n", max);
144         test_result_max_vozmojnoe(8157, max);
145 
146 }
147 
148 void test_result_max_vozmojnoe(int expected, int actual){
```

```
149     if (expected == actual)
150         puts("Ok");
151     else
152         puts("Test fail");
153 }
154
155 void automate_test_zamena_elementov_mass(){
156    test_poisk_zamena_elementov_mass();
157 }
158
159 void test_poisk_zamena_elementov_mass(){
160
161    FILE *mf;
162     mf=fopen("zamena.txt","r");
163     int n, i=0;
164     float *p;
165    fscanf(mf,"%d \n",&n);
166    p = (float *) malloc(n*sizeof(float));
167
168    for (i = 0; i<=(n-1); i++){
169        fscanf(mf,"%f\n",&p[i]);
170     }
171
172     fclose(mf);
173
174     p[0]=(p[0]+p[1])/2;
175     p[n-1]=(p[n-2]+p[n-1])/2;
176
177     for (i = 1; i<(n-1); i++){
178      p[i]=(p[i-1]+2*p[i]+p[i+1])/4;
179     }
180    for (i = 0; i<=(n-1); i++){
181    test_result_zamena_elementov_mass(p[i], p[i]);
182    }
183
184 }
185
186 void test_result_zamena_elementov_mass(int expected, int
    actual){
187     if (expected == actual)
188         puts("Ok");
189     else
190         puts("Test fail");
191 }
```

```
1 #ifndef TEST_H
2 #define TEST_H
3
4 void automate_test();
```

```
5  void test_fact_function();
6  void test_strange_function();
7  void test_result(int, int);
8  void automate_test_variant7_1();
9  void test_poisk_variant7_1();
10 void test_result_variant7_1(int, int);
11 void poisk_variant7_1();
12 void automate_test_variant7_2();
13 void test_poisk_variant7_2();
14 void test_result_variant7_2(int, int);
15 void automate_test_max_vozmojnoe();
16 void test_poisk_max_vozmojnoe();
17 void test_result_max_vozmojnoe(int, int);
18 void test_poisk_zamena_elementov_mass();
19 void automate_test_zamena_elementov_mass();
20 void test_result_zamena_elementov_mass(int, int);
21
22 #endif // TEST_H
```