Программирование

К. С. Бойцов

30 декабря 2015 г.

Глава 1

Основные конструкции языка

1.1 Задание 1

1.1.1 Задание

Пользователь задает длину отрезка в метрах. Вывести длину того же отрезка в саженях, аршинах и вершках.

1.1.2 Теоретические сведения

Воспользуемся следующим условием: `1 сажень = 3 аршина = 48 вершков, 1 вершок = 4.445 см.`

Для реализации данного алгоритма были использованы функции стандартной библиотеки, прототипы которых находятся в файле stdio.h, для ввода и вывода информации и math.h для выполнения необходимых вычислений.

1.1.3 Проектирование

В ходе проектирования было решено выделить 3 функции:

- void ui_ meter_ to_ sazhen()

  Пользователь вводит действительное `double` число метров, после чего вызывается конвертирующая функция.

- void meter_ to_ sazhen(double length)

  Функция, конвертирующая метры в сажени, аршины и вершки и выводящая результат в консоль. Параметром функции является длина отрезка в метрах типа `double`.

- void first_ task_ text()

  Функция выводит текст задания в консоль в меню пользовательского взаимодействия.

1.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор gcc (Ubuntu 5.2.1-22ubuntu2) 5.2.1 20151010, операционная система Ubuntu 15.10.

Для тестирования работы программы были выполнены статический анализ, также было проведено автоматической тестирование.

1.1.5 Тестовый план и результаты тестирования

Для статического анализа использовалась утилита Cppcheck.

Cppcheck выдал незначительные предупреждения.

При автоматическом тестировании вызывалась функция meter_ to_ sazhen, затем полученные значения сравнивались с ожидаемыми значениями. Результаты тестирования представлены в листингах.

1.1.6 Выводы

При выполнении задания я отработал свои навыки в работе с основными конструкциями языка и получил опыт в организации функций одной программы.

Листинги

meter_ to_ sazhen.c

```
1  #include "meter_to_sazhen.h"
2
3  void meter_to_sazhen(double length)
4  {
5      double sajen, arshin, vershok;
6      int int_arshin, int_sajen;
7
8      vershok = length / (4.445 * 0.01);
9      arshin = vershok / 16;
10     int_arshin = (int)arshin;
```

```
11      vershok = vershok - int_arshin * 16;
12      sajen = int_arshin / 3;
13      int_sajen = (int)sajen;
14      int_arshin = int_arshin - int_sajen * 3;
15      printf("%d sajen(s), %d arshin(s), %.1f vershok(s)\n",
            int_sajen, int_arshin, vershok);
16  }
```

ui_ meter_ to_ sazhen.c

```
1   #include "meter_to_sazhen.h"
2
3   void ui_meter_to_sazhen()
4   {
5       double length;
6       printf("\nInput the length in meters: ");
7       scanf("%lf", &length);
8       meter_to_sazhen(length);
9   }
10
11  void first_task_text()
12  {
13      printf("\nПользователь задает длину отрезка в метрах (нап
            ример, 19).\n Вывести длину того же отрезка в саженях,
             аршинах и вершках \n(например, 8 саженей 2 аршина
            11.4 вершка).\n1 сажень = 3 аршина = 48 вершков, 1 вер
            шок = 4.445 см.\n");
14  }
```

1.2 Задание 2

1.2.1 Задание

Определить, пройдет ли кирпич со сторонами , b, c сквозь прямоугольное отверстие в стене со сторонами r и s. Стороны отверстия должны быть параллельны граням кирпича.

1.2.2 Теоритические сведения

В ходе выполения задания использовалась конструкция if...else. Кроме того, были применены функции стандартной библотеки из заголовочного файла stdio.h для ввода и вывода информации.

1.2.3 Проектирование

В ходе проектирования были выделены 5 функций:

- void input_ abc(int * a, int * b, int * c)

  Параметрами функции являются 3 целых значения. Пользователю предлагается ввести 3 измерения кирпича.

- void input_ hole(int * r, int * s)

  Параметрами функции являются 2 целых значения. Пользователю предлагается ввести 2 измерения отверстия.

- int brick(int length, int width, int height, int hole_ length, int hole_ width)

  Функция имеет 5 параметров: измерения кирпича и измерения отверстия. В функции происходят сравнения величин, после чего она возвращает либо удовлетворяющий набор условий, либо неудовлетворяющий.

- void ui_ brick()

  Функция в зависимости от возвращенного значения функции `brick` выводит в консоль ответ на заданный вопрос.

- void second_ task_ text()

  Функция выводит текст задания в консоль в меню пользовательского взаимодействия.

1.2.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор gcc (Ubuntu 5.2.1-22ubuntu2) 5.2.1 20151010, операционная система Ubuntu 15.10.

Для тестирования работы программы был выполнены статический анализ.

1.2.5 Тестовый план и результаты тестирования

Для статического анализа использовалась утилита Cppcheck.

Cppcheck выдал незначительные предупреждения.

1.2.6 Выводы

При выполнении задания я получил опыт в организации функций одной программы.

Листинги

brick.c

```
#include "brick.h"

int brick(int length, int width, int height, int hole_lenght,
    int hole_width)
{
    return (((length < hole_lenght) && (width < hole_width))
       | ((width < hole_lenght) && (length < hole_width)) |
       ((length < hole_lenght) && (height < hole_width)) |
            ((height < hole_lenght) && (length < hole_width))
                | ((width < hole_lenght) && (height <
               hole_width)) | ((height < hole_lenght) && (
               width < hole_width)));
}
```

ui_ brick.c

```
#include "ui_brick.h"
#include "brick.h"

void ui_brick()
{
    int length, width, height;
    input_abc(&length, &width, &height);
    int hole_lenght, hole_width;
    input_hole(&hole_lenght, &hole_width);
    if (brick(length, width, height, hole_lenght, hole_width)
       )
        printf("Success.\n");
    else
        printf("Unsuccess.\n");
}

void input_abc(int * a, int * b, int * c)
{
    printf("Input 3 dimensions of the brick:\n");
    scanf("%d %d %d", a, b, c);
}

void input_hole(int * r, int * s)
```

```
23 {
24     printf("Input 2 dimensions of the hole:\n");
25     scanf("%d %d", r, s);
26 }
27
28 void second_task_text()
29 {
30     printf("\nОпределить, пройдет ли кирпич со сторонами a, b
           , c \nсквозь прямоугольное отверстие в стене со сторон
           ами r и s. \nСтороны отверстия должны быть параллельны
            граням кирпича.\n");
31 }
```

Глава 2

Циклы

2.1 Задание 1

2.1.1 Задание

Поменять порядок цифр заданного натурального числа на обратный.Пример: 7283916 > 6193827.Строковые функции не использовать.

2.1.2 Теоритические сведения

В ходе выполения задания для произведения необходимых вычислений и преобразований использовались операции деление "\"и деление с остатком "%". Также использовался цикл while. Кроме того, были применены функции стандартной библотеки из заголовочного файла stdio.h для ввода и вывода информации.

2.1.3 Проектирование

В ходе проектирования были выделены 3 функции:

- void ui_ swapper()

  Функция считывает из консоли целое число, которое нужно развернуть, вызывает разворачивающую функцию и выводит готовое число.

- int swapper(int number_ to_ swap)

  Параметром функции является целое число, которое нужно развернуть. В функции происходит разворот числа и возвращение его значения.

- void third_ task_ text()

  Функция выводит текст задания в консоль в меню пользовательского взаимодействия.

2.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор gcc (Ubuntu 5.2.1-22ubuntu2) 5.2.1 20151010, операционная система Ubuntu 15.10.

Для тестирования работы программы были выполнены статический анализ, также было проведено автоматическое тестирование.

2.1.5 Тестовый план и результаты тестирования

Для статического анализа использовалась утилита Cppcheck.

Cppcheck выдал незначительные предупреждения.

В ходе автоматического тестирования вызывалась функция swapper. Результаты тестирования предоставлены в листингах.

2.1.6 Выводы

В ходе выполнения я отработал навыки работы с циклами.

Листинги

swapper.c

```
#include "swapper.h"

int swapper(int number_to_swap)
{
    int swapped_number = 0;
    while ( number_to_swap != 0)
    {
        swapped_number *= 10;
        swapped_number += number_to_swap % 10;
        number_to_swap = number_to_swap / 10;
    }

    return swapped_number;
```

```
}
```

ui_ swapper.c

```
#include "ui_swapper.h"
#include "swapper.h"

void ui_swapper()
{
    int number_to_swap;
    printf("\nInput the number to turn over: ");
    scanf("%d", &number_to_swap);
    printf("\nThe turnover: %d\n\n", swapper(number_to_swap))
        ;
}

void third_task_text()
{
    printf("\nПоменять порядок цифр заданного натурального чи
        сла на обратный.\nПример: 7283916 > 6193827.\nСтроковы
        е функции не использовать.\n");
}
```

Глава 3

Массивы

3.1 Задание 1

3.1.1 Задание

В матрице $X(m, n)$ каждый элемент (кроме граничных) заменить суммой непосредственно примыкающих к нему элементов по вертикали, горизонтали и диагоналям.

3.1.2 Теоритические сведения

Для выполнения задания использовался цикл for, конструкция if...else, а также функции стандартной библиотеки из заголовочного файла stdlib.h для динамического выделения и освобождения памяти и stdio.h для ввода, вывода информации и работы с файлами.

3.1.3 Проектирование

Ввод и вывод данных реализован с помощью файлов. Входной файл должен содержать матрицу из целых чисел.

В ходе проектирования были выделены следующие функции:

- void ui_ matrix()

  Пользователь вводит нужное ему количество столбцов и рядов из исходной матрицы, после чего создаётся и выводится в консоль созданная матрица.

- void matrix(int coloumns, int rows)

Параметрами функции являются целые количества столбцов и рядов. В функции выделяется память, генерируются и выводятся исходная и новая матрицы, после чего память освобождается.

- void fourth_ task_ text()

  Функция выводит текст задания в консоль в меню пользовательского взаимодействия.

3.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор gcc (Ubuntu 5.2.1-22ubuntu2) 5.2.1 20151010, операционная система Ubuntu 15.10.

Для тестирования работы программы был выполнены статический анализ.

3.1.5 Тестовый план и результаты тестирования

Для статического анализа использовалась утилита Cppcheck.

Cppcheck выдал незначительные предупреждения.

3.1.6 Выводы

При выполнении задания я понял принцип организации программы при работе с выделением динамической памяти, научился работать с файлами.

Листинги

matrix.c

```
1  #include "matrix.h"
2
3  void matrix(int coloumns, int rows)
4  {
5      FILE *myfile = fopen("myfile", "r");
6      if (myfile == NULL)
7      {
8          puts("Error file!");
9          exit(1);
10     }
11
```

```
12      int * matrix[rows];
13      int i;
14      for (i = 0; i < rows; i++)
15      {
16          matrix[i] = (int*)malloc(sizeof(int)*coloumns);
17      }
18
19      int m, n;
20      for(m = 0; m < rows; m++)
21      {
22          for (n = 0; n < coloumns; n++)
23          {
24              fscanf(myfile, "%d", &matrix[m][n]);
25              printf("%2d ", matrix[m][n]);
26          }
27      printf("\n");
28      }
29      printf("\n");
30
31      int * post_matrix[rows];
32      for (i = 0; i < rows; i++)
33      {
34          post_matrix[i] = (int*)malloc(sizeof(int)*coloumns);
35      }
36
37
38      for(m = 0; m < rows; m++)
39      {
40          for (n = 0; n < coloumns; n++)
41          {
42              if ((m == 0) || (m == coloumns-1) || (n == 0) ||
                    (n == coloumns-1))
43                      post_matrix[m][n] = matrix[m][n];
44              else
45                      post_matrix[m][n] = matrix[m-1][n-1] +
                            matrix[m-1][n] + matrix[m][n-1] +
                            matrix[m-1][n+1] + matrix[m+1][n-1] +
                            matrix[m+1][n] + matrix[m][n+1] +
                            matrix[m+1][n+1];
46              printf("%2d ", post_matrix[m][n]);
47          }
48          printf("\n");
49      }
50
51      fclose(myfile);
52      for (i = 0; i < rows; i++)
53      {
54          free(matrix[i]);
55          free(post_matrix[i]);
```

```
56      }
57
58      free(matrix);
59      free(post_matrix);
60  }
```

ui_ matrix.c

```
1   #include "ui_matrix.h"
2   #include "matrix.h"
3
4   void ui_matrix()
5   {
6       int coloumns, rows;
7       puts("Enter coloumns: ");
8       scanf("%d", &coloumns);
9       puts("Enter rows: ");
10      scanf("%d", &rows);
11      printf("\n");
12      matrix(coloumns, rows);
13  }
14
15  void fourth_task_text()
16  {
17      printf("\nВ матрице X(m,n) каждый элемент (кроме граничны
            x) \nзаменить суммой непосредственно примыкающих к нем
            y \nэлементов по вертикали, горизонтали и диагоналям.\
            n");
18  }
```

Глава 4

Строки

4.1 Задание 1

4.1.1 Задание

Текст, не содержащий собственных имен и сокращений, набран полностью прописными русскими буквами.Заменить все прописные буквы, кроме букв, стоящих после точки, строчными буквами.

4.1.2 Теоритические сведения

Для выполнения задания использовался цикл for, конструкция if...else, а также функции стандартной библиотеки из заголовочного файла ctype.h для работы со строками и stdio.h для ввода, вывода информации.

4.1.3 Проектирование

В ходе проектирования были выделены следующие функции:

- void ui_ lower()

  Инициализируется массив из символов, после чего поочерёдно вызываются две следующие функции и выводится конечный ответ.

- void input_ lower(char * str)

  Пользователю предлагается ввести строку, в которой все буквы будут прописными.

- void lower_ case(char * str)

Все прописные буквы в предложении, кроме первой,заменяются на строчные.

- void fifth_ task_ text()

  Функция выводит текст задания в консоль в меню пользовательского взаимодействия.

4.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор gcc (Ubuntu 5.2.1-22ubuntu2) 5.2.1 20151010, операционная система Ubuntu 15.10.

Для тестирования работы программы был выполнены статический анализ.

4.1.5 Тестовый план и результаты тестирования

Для статического анализа использовалась утилита Cppcheck.

Cppcheck не выдал ошибок.

4.1.6 Выводы

При выполнении задания я научился пользоваться функциями для работы со строками.

Листинги

lower.c

```
1  #include "lower.h"
2
3  void lower_case(char * str)
4  {
5      int i;
6      for (i = 1; str[i] != '\0'; i++)
7      {
8          if (str[i] == '.')
9          {
10             i++;
11             if (str[i] == ' ')
12                 i++;
```

```
13          }
14          else
15              str[i] = tolower(str[i]);
16      }
17  }
```

ui_ lower.c

```
1  #include "ui_lower.h"
2  #include "lower.h"
3
4  void ui_lower()
5  {
6      char str[100];
7      input_lower(str);
8      lower_case(str);
9      printf("%s", str);
10 }
11
12 void input_lower(char * str)
13 {
14     puts("\nInput string: ");
15     scanf("%*c");
16     gets(str);
17 }
18
19 void fifth_task_text()
20 {
21     printf("\nТекст, не содержащий собственных имен и сокраще
           ний, \nнабран полностью прописными русскими буквами.\n
           Заменить все прописные буквы, кроме букв, \nстоящих по
           сле точки, строчными буквами.\n");
22 }
```

Глава 5

Инкапсуляция

5.1 Задание 1

5.1.1 Задание

Реализовать класс РАЦИОНАЛЬНОЕ ЧИСЛО (представимое в виде m/n). Требуемые методы: конструктор, деструктор, копирование, сложение, вычитание, умножение, деление, преобразование к типу double.

5.1.2 Теоритические сведения

Для выполнения задания использовался цикл for, конструкция if...else, а также класс exception стнадартной бибилиотеки.

5.1.3 Проектирование

В ходе проектирования программы было решено создать класс, который называется RationalNum. Созданный класс содержит 2 поля с модификатором доступа private:

- int numerator;
- int denominator; Числитель и знаменатель рационального числа.

В классе определен конструктор

- RarionalNum(int numerator = 1, int denominator = 8)
  Конструктор со значениями по умолчанию для числа.

В классе определены 5 методов с модификатором доступа public:

1. void Copy(RationalNum);

   Метод, аналогичный конструктору копирования.

2. void Sum(int);

   Метод обеспечивает сложение.

3. void Multi(int);

   Метод обеспечивает умножение.

4. void Divide(int);

   Метод обеспечивает деление.

5. double ToDouble();

   Метод преобразует рациональное число к типу double. Возвращает соответственно это число.

   Так же перегружены операторы сложения, умножения и деления.

Так же был создаы класса исключений:

- DevNull Исключение вызывается, когда совершается попытка деления на ноль.

5.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор gcc (Ubuntu 5.2.1-22ubuntu2) 5.2.1 20151010, операционная система Ubuntu 15.10.

Для тестирования работы программы были выполнены статический анализ, также было проведено автоматической тестирование.

5.1.5 Тестовый план и результаты тестирования

Для статического анализа использовалась утилита Cppcheck.

Cppcheck ошибок не обнаружил.

Результаты автоматического тестирования представлены в листингах.

5.1.6 Выводы

При выполнении задания я понял принцип инкапсуляции и организации полей и методов класса.

Листинги

rat_ num.h

```
#ifndef RAT_NUM_H
#define RAT_NUM_H


#include <iostream>
#include <exception>

using namespace std;

class RationalNum
{
    int numerator;
    int denominator;
public:
    RationalNum(int numerator = 1, int denominator = 8);
    void Copy(RationalNum);
    void Sum(int);
    void Multi(int);
    void Divide(int);
    double ToDouble();
    RationalNum operator+(int);
    RationalNum operator*(int);
    RationalNum operator/(int);

private:

};
class DevNull:public exception{
public:


};

#endif // RAT_NUM_H
```

rat_ num.cpp

```
#include "rat_num.h"

```

```
3
4  RationalNum::RationalNum(int numerator, int denominator):
       numerator(numerator), denominator(denominator){}
5
6  void RationalNum::Copy(RationalNum numb)
7  {
8      numerator = numb.numerator;
9      denominator = numb.denominator;
10 }
11
12 void RationalNum::Sum(int num)
13 {
14     numerator += num*denominator;
15
16 }
17
18 void RationalNum::Multi(int num)
19 {
20     numerator *= num;
21
22 }
23
24 void RationalNum::Divide(int num)
25 {
26     if (num == 0){
27         DevNull error;
28         throw error;
29     }
30     denominator *= num;
31
32 }
33
34 RationalNum RationalNum::operator/(int num)
35 {
36     if (num == 0){
37         DevNull error;
38         throw error;
39     }
40     RationalNum n;
41     n.denominator *= num;
42     return n;
43 }
44
45 double RationalNum::ToDouble(){
46     return((double)numerator / double(denominator));
47 }
48
49 RationalNum RationalNum::operator+(int num)
50 {
```

```
51      RationalNum n;
52      n.numerator += num*n.denominator;
53      return n;
54  }
55
56  RationalNum RationalNum::operator*(int num)
57  {
58      RationalNum n;
59      n.numerator *= num;
60      return n;
61  }
```

Глава 6

Приложение

6.1 Автоматические тесты

```
#include <QString>
#include <QtTest>
#include "rat_num.h"

using std::string;

class CpptestTest : public QObject
{
    Q_OBJECT

public:
    CpptestTest();

private Q_SLOTS:
    void test_rational_sum();
    void test_rational_multi();
    void test_rational_divide();
};

CpptestTest::CpptestTest()
{

}

void CpptestTest::test_rational_divide(){
    RationalNum num;
    num.Divide(2);
    QCOMPARE(num.ToDouble(), 0.06250);
}
void CpptestTest::test_rational_multi(){
```

```
    RationalNum num;
    num.Multi(2);
    QCOMPARE(num.ToDouble(), 0.25);
}

void CpptestTest::test_rational_sum(){
    RationalNum num;
    num.Sum(1);
    QCOMPARE(num.ToDouble(), 1.125);
}

QTEST_APPLESS_MAIN(CpptestTest)

#include "tst_cpptesttest.moc"
```

```
#include <QString>
#include <QtTest>
#include "meter_to_sazhen.h"
#include "swapper.h"

class BoitsovTest : public QObject
{
    Q_OBJECT

public:
    BoitsovTest();

private Q_SLOTS:
    void test_meter_to_sazhen();
    void test_swapper();
};

BoitsovTest::BoitsovTest()
{
}

void BoitsovTest::test_meter_to_sazhen()
{
    double length = 19;
    int int_sazhen, int_arshin;
    double vershok;
    meter_to_sazhen(length);
    QCOMPARE(int_sazhen, 8);
    QCOMPARE(int_arshin, 7);
    QCOMPARE(vershok, 11.4);
}

void BoitsovTest::test_swapper()
{
```

```
35      int number_to_swap = 54321;
36      int swapped_number;
37      swapper(number_to_swap);
38      QCOMPARE(swapped_number, 12345);
39  }
40
41  QTEST_APPLESS_MAIN(BoitsovTest)
42
43  #include "tst_testtesttest.moc"
```