Программирование

А. А. Ильин

25 декабря 2015 г.

Глава 1

Основные конструкции языка

1.1 Задание 1

1.1.1 Задание

Пользователь задает угол в градусах, минутах и секундах. Вывести значение того же угла в радианах.

1.1.2 Теоретические сведения

Радиан - радианная мера угла. Радиан связан с градусами следующим соответствием:

$$1 radian = 180/\pi degrees$$

Градусы в свою очередь делятся на секунды и минуты: $1 degree = 60 min$ $1 min = 60 sec$

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

Для более удобного хранения данных, а так же их передачи была использована структура Angle. Она содержит 3 поля, которые должны содержать целые числа, соответсвующие градусам, минутам и секундам.

В ходе проектирования было решено выделить одну функцию:

- void translation(double, Angle*)

Функция вычисляет переводит из радиан в градусы. Параметрами функциия являются переменная типа double и указатель на созданную структуру. Первое значение соответствует числу радиан, переданному пользователем, а в структуру на которую получен указетль будет записаны градусы, минуты и секунды.

1.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор gcc (Debian 4.9.2-10) 4.9.2, операционная система Linux version 3.16.0-4-586.

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, затем полученные значения сравнивались с ожидаемыми значениями. Результаты тестирования представлены в листингах.

1.1.6 Выводы

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

Листинги

translation.c

```
1 #include "translation.h"
2
3
4 void translation(double radian, Angle* angle)
5 {
6
7     double tmp = radian * 180 / 3.14;
8     angle->degree = floor(tmp);
```

```
9
10      tmp = (tmp-floor(tmp)) * 60;
11      angle->min = floor(tmp);
12
13      tmp = (tmp - floor(tmp)) * 60;
14      angle->sec = floor(tmp);
15
16
17  }
```

quadEquationUI.c

```
1  #include "ui_translation.h"
2  #include "translation.h"
3
4  void ui_translation()
5  {
6      Angle angle;
7      double radian;
8      printf("Input the angle in radians:\n");
9      scanf("%lf", &radian);
10     translation(radian, &angle);
11     printf("Degree: %d, Minutes: %d, Seconds: %d\n", angle.
           degree, angle.min, angle.sec);
12 }
```

1.2 Задание 2

1.2.1 Задание

Мой возраст. Для заданного N рассматриваемого как возраст человека, вывести фразу вида: «Мне 21 год», «Мне 32 года», «Мне 12 лет».

1.2.2 Теоритические сведения

В ходе выполения задания для произведения необходимых вычислений и преобразований использовались перечисляемый тип и деление с остатком "%". Также 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 следущая функция:

- int tell_ me_ age(int)

  Функция проверяет число на несколько условий и возвращает один из идентефикаторов. Перечилсяемый тип был использован для систематизации вывода информации.

1.2.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор gcc (Debian 4.9.2-10) 4.9.2, операционная система Linux version 3.16.0-4-586.

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

tell_ me_ age.c

```
1  #include "tell_me_age.h"
2
3  int tell_me_age(int age){
4
5      enum years {Ages, Year, Years};
6      if
7              ((age >= 11) && (age <= 14) )
8          return Ages;
9      if ( ((age % 10) < 5) && ((age % 10 ) > 1) )
10         return Year;
11     if ((age % 10) == 1)
12         return Years;
13     return Ages;
14 }
```

ui_ tell_ me_ age.c

```c
#include "ui_tell_me_age.h"
#include "tell_me_age.h"

void ui_tell_me_age(){
    int age;
    printf("Input your age.\n");
    scanf("%d", &age);
    int ans = tell_me_age(age);
    switch(ans){
    case 0: printf("Вам %d лет!", age); break;
    case 1: printf("Вам %d года!", age); break;
    case 2: printf("Вам %d год!", age); break;
    }
}
```

Глава 2

Циклы

2.1 Задание 1

2.1.1 Задание

Найти число, полученное из данного дублированием четных цифр.

2.1.2 Теоритические сведения

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

2.1.3 Проектирование

В ходе проектирования были выделены следущая функция:

- int double_ even_ numbers(int)

  Функция ищет число цифр в числе, проверяет цифру на четность и в случае истинности дублирует ее.

2.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор gcc (Debian 4.9.2-10) 4.9.2, операционная система Linux version 3.16.0-4-586.

Для тестирования работы программы были выполнены статический анализ, также было проведено автоматической 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 double_ even_ numbers. Результаты тестирования предоставлены в листингах.

2.1.6 Выводы

В ходе выполнения я отработал навыки работы с циклами.

Листинги

double_ even_ numbers.c

```c
#include"double_even_numbers.h"

int double_even_numbers(int input_num){


    int output_num = 0, tmp, amount_of_numerals;
    tmp = input_num;

    for (amount_of_numerals=0; tmp > 0; amount_of_numerals++)
        {
        tmp = tmp / 10;
    }

    while(input_num > 0){
        tmp = input_num / pow(10, amount_of_numerals-1);
        int remaining_power = pow(10, amount_of_numerals-1);

        if (tmp % 2 == 0)
            output_num = output_num * 100 + tmp * 10 + tmp;
        else
            output_num = output_num * 10 + tmp;

        input_num = input_num % remaining_power;
        amount_of_numerals--;
    }
```

```
25      return output_num;
26  }
```

ui_ double_ even_ numbers.c

```
1  #include"ui_conversation.h"
2  #include "double_even_numbers.h"
3
4  void ui_conversation(){
5      int input_num;
6      printf("Input number:\n");
7      scanf("%d", &input_num);
8      printf("%d", double_even_numbers(input_num));
9  }
```

Глава 3

Массивы

3.1 Задание 1

3.1.1 Задание

Удалить из массива $A(n)$ нулевые элементы, передвинув на их место следующие элементы без нарушения порядка их следования. В результате должен получиться массив меньшего размера, не содержащий нулей.

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, stdio.h для ввода, вывода информации и работы с файлами и math.h для выполнения вычислений.

3.1.3 Проектирование

Ввод и вывод данных реализован с помощью файлов. Входной файл должен содержать некоторое количество целых чисел, записанных через пробел. Выходные файлы создаются по ходу программы и также содержат целые числа, записанные через пробел.

В ходе проектирования были выделены следующие функции:

- int array_ not_ zero(int*, int) Функция получает массив целых чисел, считанный из файла, а так же его размер. Затем по циклу ищет нули и удаляет их с массива, затем возвращает индекс последнего элемента массива для последующего вывода.

3.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор gcc (Debian 4.9.2-10) 4.9.2, операционная система Linux version 3.16.0-4-586.

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

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

array_ without_ nulls.c

```
#include"array_without_nulls.h"

int array_not_zero(int* array, int size){
    int k, i;
       for (i=0; i<size; i++){

        for (k=1; k<size-i; k++){

            if (array[i] == 0){

                array[i] = array[i+k];
                array[i+k] = 0;

            }
            else break;
        }
    }

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

        if (array[i] == 0) {k = i;
```

```
            break;
        }

    }
    return k;
      }
```

ui_ array_ without_ nulls.c

```
#include "ui_array.h"
#include"array_without_nulls.h"
#include<stdlib.h>
void ui_array_without_nulls(){
    int size = 20;
    int* array = (int*) malloc(sizeof(int)*size) ;
    int i;

    FILE *myfile = fopen("myfile.txt", "r");
    for(i = 0; i< size; i++)
        fscanf(myfile, "%d", &array[i]);

    fclose(myfile);

    int k = array_not_zero(array, size);

    FILE *output = fopen("Output.txt", "w");
    int* new_array = (int*) malloc(sizeof(int)*20);
    for (i=0; i<k; i++){

        new_array[i] = array[i];
        fprintf(output,"%d ", new_array[i]);

    }
    fclose(output);
    free(array);
    free(new_array);
    printf("Done.\n");
}
```

Глава 4

Строки

4.1 Задание 1

4.1.1 Задание

В русском языке, как правило, после букв Ж, Ч, Ш, Щ пишется И, А, У, а не Ы, Я, Ю. Проверить заданный текст на соблюдение этого правила и исправить ошибки.

4.1.2 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, stdio.h для ввода, вывода информации и работы с файлами и string.h для работы со строками.

4.1.3 Проектирование

В ходе проектирования были выделены следующие функции:

- void check_ sizzling(char*, int) Функция получает строку и ее размер, затем в строке выполняется поиск шипящей согласной и перенаправление на проверку последующей буквы, и при необходимости ее изменение.
- int check_ vowel(char) Функция проверяет на ошибку следующую после шипящей букву, и возвращает соответствующее значение.

4.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор gcc (Debian 4.9.2-10) 4.9.2, операционная система Linux version 3.16.0-4-586.

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

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 отработал навыки работы с файлами и научился пользоваться функциями для работы со строками.

Листинги

check_ sizzling.c

```
1
2  #include"check_sizzling.h"
3
4  void check_sizzling(char* string, int size){
5      int i;
6      for(i=0; i<size; i++)
7          if (string[i] == 'r' || string[i] == 'n'||string[i]
               == 'v')
8              switch(check_vowel(string[i+1])){
9              case 0:
10                 string[i+1] = 'a';
11                 break;
12             case 1:
13                 string[i+1] = 'u';
14                 break;
15             case 2:
16                 string[i+1] = 'e';
17                 break;
18             default: break;
19
20             }
```

```
21
22 }
23 int check_vowel(char vowel){
24     if(vowel == 'e') return 0;
25     if(vowel == 'i') return 1;
26     if(vowel == 'y') return 2;
27     return 3;
28
29 }
```

ui_ check_ sizzling.c

```
1
2 #include "ui_check_sizzling.h"
3 #include"check_sizzling.h"
4 #include <stdlib.h>
5 #include<stdio.h>
6
7 void ui_check_sizzling(){
8     int size = 20;
9     char* string = (char*) malloc(size);
10    FILE * textfile = fopen("somefile", "r");
11           fgets(string,size, textfile);
12    fclose(textfile);
13    check_sizzling(string, size);
14    FILE * outfile = fopen("outfile", "w");
15    fprintf(outfile,"%s", string);
16    fclose(outfile);
17 }
```

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 (Debian 4.9.2-10) 4.9.2, операционная система Linux version 3.16.0-4-586.

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

rationalnum.h

```
#ifndef RATIONALNUM_H
#define RATIONALNUM_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 // RATIONALNUM_H
```

rationalnum.cpp

```
#include "rationalnum.h"


RationalNum::RationalNum(int numerator, int denominator):
   numerator(numerator), denominator(denominator){}
```

```
5
6  void RationalNum::Copy(RationalNum numb){
7      numerator = numb.numerator;
8      denominator = numb.denominator;
9  }
10 void RationalNum::Sum(int num){
11     numerator += num*denominator;
12
13 }
14 void RationalNum::Multi(int num){
15     numerator *= num;
16
17 }
18 void RationalNum::Divide(int num){
19     if (num == 0){
20         DevNull error;
21         throw error;
22     }
23     denominator *= num;}
24
25 RationalNum RationalNum::operator/(int num){
26     if (num == 0){
27         DevNull error;
28         throw error;
29     }
30     RationalNum n;
31     n.denominator *= num;
32     return n;
33 }
34
35 double RationalNum::ToDouble(){
36     return((double)numerator / double(denominator));
37 }
38
39 RationalNum RationalNum::operator+(int num)
40 {
41     RationalNum n;
42     n.numerator += num*n.denominator;
43     return n;
44 }
45
46 RationalNum RationalNum::operator*(int num)
47 {
48     RationalNum n;
49     n.numerator *= num;
50     return n;
51 }
```

Глава 6

Приложение

6.1 Классы для реализации заданий 1-5

1. Класс translation, перевод радиан.

translation.h

```
#ifndef TRANSLATION_H
#define TRANSLATION_H
#include <cmath>

class Translation
{
private:
    double radian;
    int sec;
    int min;
    int degree;

public:
    Translation(const double rad = 0.5);
    void convert();
    int getSec(Translation &) const;
    int getMin(Translation &) const;
    int getDegree(Translation &) const;
};

#endif // TRANSLATION_H
```

translation.cpp

```
#include "translation.h"

Translation::Translation(const double rad)
```

```
{
    radian = rad;
    degree = 0;
    min = 0;
    sec = 0;
}

void Translation::convert(){
    double tmp = radian * 180 / 3.14;
    degree = floor(tmp);

    tmp = (tmp-floor(tmp)) * 60;
    min = floor(tmp);

    tmp = (tmp - floor(tmp)) * 60;
    sec = floor(tmp);
}

int Translation::getSec(Translation &) const
{
 return sec;
}

int Translation::getMin(Translation &) const
{
    return min;
}

int Translation::getDegree(Translation &) const
{
    return degree;
}
```

2. Класс tell_ me_ age манипуляции с возрастом.

   tell_ me_ age.h

```
#ifndef TELL_ME_AGE_H
#define TELL_ME_AGE_H
#include<string>
using std::string;


class tell_me_age
{
private:
    int age;
public:
    tell_me_age(int input = 12);
```

```
13      int checking_age() const;
14      string text_to_out(const int) const;
15  };
16
17  #endif // TELL_ME_AGE_H
```

tell_ me_ age.cpp

```
1   #include "tell_me_age.h"
2
3   tell_me_age::tell_me_age(int input)
4   {
5       age = input;
6   }
7
8   int tell_me_age::checking_age() const
9   {
10      enum years {Ages, Year, Years};
11      if
12              ((age >= 11) && (age <= 14) )
13          return Ages;
14      if ( ((age % 10) < 5) && ((age % 10 ) > 1) )
15          return Year;
16      if ((age % 10) == 1)
17          return Years;
18      return Ages;
19  }
20
21  std::string tell_me_age::text_to_out(const int years)
        const
22  {
23      switch(years){
24      case 0: return ("лет");
25      case 1: return ("года");
26      case 2: return ("год");
27      }
28      return ("этого не будет все равно");
29  }
```

3. Класс double_ even_ numbers дублирование четных цифр

double_ even_ numbers.h

```
1   #ifndef CONVERSATION_H
2   #define CONVERSATION_H
3   #include<cmath>
4
5   class double_even_numbers
```

```
{
private:

public:
    int convert(const int input_num = 1234) const;
};

#endif // CONVERSATION_H
```

double_ even_ numbers.cpp

```
#include "double_even_numbers.h"

int double_even_numbers::convert(int input_num) const
{   int output_num = 0;
    int tmp, amount_of_numerals;
    tmp = input_num;

    for (amount_of_numerals=0; tmp > 0;
        amount_of_numerals++){
        tmp = tmp / 10;
    }

    while(input_num > 0){
        tmp = input_num / pow(10, amount_of_numerals-1);
        int remaining_power = pow(10, amount_of_numerals
            -1);

        if (tmp % 2 == 0)
            output_num = output_num * 100 + tmp * 10 +
                tmp;
        else
            output_num = output_num * 10 + tmp;

        input_num = input_num % remaining_power;
        amount_of_numerals--;
    }
    return output_num;
}
```

4. Класс array для инициализации массива и удаления из него нулей.

array.h

```
#ifndef MATRIX_WITHOUT_NULLS_H
#define MATRIX_WITHOUT_NULLS_H

class Array
```

```
5  {
6  private:
7      int size;
8      int * pointer;
9  public:
10     Array(int size = 15);
11     Array(Array &);
12     void operator=(Array &);
13     int operator[](int) const;
14     ~Array();
15     void delete_nulls(Array &);
16 };
17
18 #endif // MATRIX_WITHOUT_NULLS_H
```

array.cpp

```
1  #include "array.h"
2  #include "iostream"
3
4  Array::Array(int size)
5  {
6      this->size = size;
7      pointer = new int[this->size];
8      for (int i = 0; i < this->size; i++)
9          pointer[i] = 0;
10
11 }
12
13 Array::Array(Array & array)
14 {
15     size = array.size;
16     pointer = new int [size];
17     for (int i = 0; i < size; i++)
18         pointer[i] = array[i];
19 }
20
21
22
23 void Array::operator=(Array & array)
24 {
25     delete [] pointer;
26     size = array.size;
27     pointer = new int [size];
28     for (int i = 0; i < size; i++)
29         pointer[i] = array[i];
30 }
31
32 int Array::operator[](int i) const
```

```
33 {
34     return pointer[i];
35 }
36
37 Array::~Array()
38 {
39     delete [] pointer;
40 }
41
42 void Array::delete_nulls(Array & array)
43 {   Array tmparray;
44     tmparray = array;
45     delete [] array.pointer;
46     int k, i;
47     for (i=0; i<tmparray.size; i++){
48
49         for (k=1; k<tmparray.size-i; k++){
50
51             if (tmparray.pointer[i] == 0){
52
53                 tmparray.pointer[i] = array.pointer[i+k];
54                 tmparray.pointer[i+k] = 0;
55
56             }
57             else break;
58         }
59     }
60     for (i = 0; i < tmparray.size;i++)
61
62         if (array.pointer[i] == 0) {k = i;
63             break;
64         }
65     array.pointer = new int[k];
66
67     for (i=0; i<k; i++)
68
69         array.pointer[i] = tmparray.pointer[i];
70
71
72 }
```

5. Класс check_ sizzling для удаления шипящих.

   check_ sizzling.h

```
1 #ifndef CHECK_SIZZLING_H
2 #define CHECK_SIZZLING_H
3 #include<string>
4 using std::string;
```

```
5
6  class check_sizzling
7  {
8
9  public:
10     string find_symbol_and_change_if_need(const string
           text = "restart nimfa vywern") const;
11 private:
12     int check_symbol(const char) const;
13
14 };
15
16 #endif // CHECK_SIZZLING_H
```

check_ sizzling.cpp

```
1  #include "check_sizzling.h"
2
3
4  std::string check_sizzling::
       find_symbol_and_change_if_need(string text) const
5  {
6      int i;
7      for(i=0; i<20; i++)
8          if (text[i] == 'r' || text[i] == 'n'||text[i] ==
               'v')
9              switch(check_symbol(text[i+1])){
10             case 0:
11                 text[i+1] = 'a';
12                 break;
13             case 1:
14                 text[i+1] = 'u';
15                 break;
16             case 2:
17                 text[i+1] = 'e';
18                 break;
19             default: break;
20
21             }
22     return text;
23 }
24
25 int check_sizzling::check_symbol(char vowel) const
26 {
27     if(vowel == 'e') return 0;
28     if(vowel == 'i') return 1;
29     if(vowel == 'y') return 2;
30     return 3;
31 }
```

6.2 Автоматические тесты

```
#include <QString>
#include <QtTest>
#include "check_sizzling.h"
#include"double_even_numbers.h"
#include <string>
#include "rationalnum.h"
using std::string;

class CpptestTest : public QObject
{
    Q_OBJECT




private Q_SLOTS:
    void test_conversation();
    void test_sizzling();
    void test_rational_sum();
    void test_rational_multi();
    void test_rational_divide();

};

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);
}


void CpptestTest::test_conversation(){
    double_even_numbers num;
    QCOMPARE(num.convert(), 122344);
}
void CpptestTest::test_sizzling(){
```

```
47      check_sizzling intext;
48      QVERIFY2((intext.find_symbol_and_change_if_need() == "
            rastart numfa vewern"), "Failure");
49  }
50
51  QTEST_APPLESS_MAIN(CpptestTest)
52
53  #include "tst_cpptesttest.moc"
```

```
1   #include <QString>
2   #include <QtTest>
3   #include "double_even_numbers.h"
4   #include "translation.h"
5
6
7   class IlinTest : public QObject
8   {
9       Q_OBJECT
10
11  public:
12      IlinTest();
13
14  private Q_SLOTS:
15     void test_conversation();
16     void test_translation();
17  };
18
19  IlinTest::IlinTest()
20  {
21  }
22
23  void IlinTest::test_conversation()
24  {    int number = 12234;
25          QCOMPARE(double_even_numbers(number), 12222344);
26
27  }
28  void IlinTest::test_translation()
29  {
30      double radians = 0.5;
31      Angle angle;
32      translation(radians, &angle);
33      QCOMPARE(angle.degree, 28);
34      QCOMPARE(angle.min, 39);
35      QCOMPARE(angle.sec, 44);
36
37  }
38
39  QTEST_APPLESS_MAIN(IlinTest)
40
```

```
41 #include "tst_ilintest.moc"
```