Программирование

А. А. Ильин

24 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 Qt 5.5.0 MinGW 32bit, операционная система Linux version 3.16.0-4-586.

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

1.1.5 Тестовый план и результаты тестирования

Для статического анализа использовалась утилита Cppcheck.

Результат проверки translation.c
Статистика:
Ошибки: 0
Предупреждения: 0
Стилистические предупреждения: 0
Предупреждения переносимости: 0
Предупреждения производительности: 0
Информационные сообщения: 0

При автоматическом тестировании вызывалась функция, затем полученные значения сравнивались с ожидаемыми значениями. Результаты тестирования представлены ниже:

Тестируемое значение: radians = 0.5 Ожидаемые значения: degree = 28 min = 39 sec = 44

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

```
#include "translation.h"


void translation(double radian, Angle* angle)
{

    double tmp = radian * 180 / 3.14;
    angle->degree = floor(tmp);

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

    tmp = (tmp - floor(tmp)) * 60;
    angle->sec = floor(tmp);


}
```

quadEquationUI.c

```
#include "ui_translation.h"
#include "translation.h"

void ui_translation()
{   Angle angle;
    double radian;
    printf("Input the angle in radians:\n");
    scanf("%lf", &radian);
    translation(radian, &angle);
    printf("Degree: %d, Minutes: %d, Seconds: %d\n", angle.
       degree, angle.min, angle.sec);
}
```

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 Qt 5.5.0 MinGW 32bit, операционная система Linux version 3.16.0-4-586.

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

1.2.5 Тестовый план и результаты тестирования

Для статического анализа использовалась утилита Cppcheck.

Результат проверки
Статистика:
Ошибки: 0
Предупреждения: 0
Стилистические предупреждения: 0
Предупреждения переносимости: 0
Предупреждения производительности: 0
Информационные сообщения: 0

1.2.6 Выводы

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

Листинги

tell_ me_ age.c

```c
#include "tell_me_age.h"

int tell_me_age(int age){

    enum years {Ages, Year, Years};
    if
            ((age >= 11) && (age <= 14) )
        return Ages;
    if ( ((age % 10) < 5) && ((age % 10 ) > 1) )
        return Year;
    if ((age % 10) == 1)
        return Years;
    return Ages;
}
```

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 conversation(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 Qt 5.5.0 MinGW 32bit, операционная система Linux version 3.16.0-4-586.

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

2.1.5 Тестовый план и результаты тестирования

Для статического анализа использовалась утилита Cppcheck.

Результат проверки factorization.c
Статистика:
Ошибки: 0
Предупреждения: 0
Стилистические предупреждения: 0
Предупреждения переносимости: 0
Предупреждения производительности: 0
Информационные сообщения: 0

Результат проверки factorizationUI.c
Статистика:
Ошибки: 0
Предупреждения: 0
Стилистические предупреждения: 0
Предупреждения переносимости: 0
Предупреждения производительности: 0
Информационные сообщения: 0

В ходе автоматического тестирования вызывалась функция translation, которой подавались различные числа для факторизации. Затем полученные данные сравнивались с ожидаемым результатом.

Тестируемое значение: 12234 Ожидаемое значения: 12222344

2.1.6 Выводы

В ходе выполнения я отработал навыки работы с циклами.

Листинги

conversation.c

```c
#include"conversation.h"



int conversation(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--;
    }
    return Output_num;
}
```

ui_ conversation.c

```c
#include"ui_conversation.h"
#include "conversation.h"

void ui_conversation(){
    int input_num;
    printf("Input number:\n");
    scanf("%d", &input_num);
    printf("%d", conversation(input_num));
}
```

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 matrix_ not_ zero(int*) Функция получает массив целых чисел, считанный из файла. 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 Qt 5.5.0 MinGW 32bit, операционная система Linux version 3.16.0-4-586.

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

3.1.5 Тестовый план и результаты тестирования

Для статического анализа использовалась утилита Cppcheck.

Результат проверки arrays.c
Статистика:
Ошибки: 0
Предупреждения: 0
Стилистические предупреждения: 0
Предупреждения переносимости: 0
Предупреждения производительности: 0
Информационные сообщения: 0

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
4  int matrix_not_zero(int* array){
5      int k, i;
6         for (i=0; i<20; i++){
7
8          for (k=1; k<20-i; k++){
9
10             if (array[i] == 0){
11
12                 array[i] = array[i+k];
```

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

            }
            else break;
        }
    }
    printf("\n");
    for (i = 0; i < 20;i++){

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

    }
    return k;
      }
```

ui_ matrix.c

```
#include "ui_matrix.h"
#include"matrix.h"
#include<stdlib.h>
void ui_matrix(){
    int* array = (int*) malloc(sizeof(int)*20) ;
    int i;

    FILE *myfile = fopen("myfile.txt", "r");
    for(i = 0; i< 20; i++)
        fscanf(myfile, "%d", &array[i]);

    fclose(myfile);

    int k = matrix_not_zero(array);

    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_ slizzing(char*) В этой функции выполняется поиск шипящей согласной и перенаправление на проверку последующей буквы, и при необходимости изменяет ее.
- 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 Qt 5.5.0 MinGW 32bit, операционная система Linux version 3.16.0-4-586.

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

4.1.5 Тестовый план и результаты тестирования

Для статического анализа использовалась утилита Cppcheck.

Результат проверки checkExpirationTime.c
Статистика:
Ошибки: 0
Предупреждения: 0
Стилистические предупреждения: 0
Предупреждения переносимости: 0
Предупреждения производительности: 0
Информационные сообщения: 0

Результат проверки checkExpirationTimeUI.c
Статистика:
Ошибки: 0
Предупреждения: 0
Стилистические предупреждения: 0
Предупреждения переносимости: 0
Предупреждения производительности: 0
Информационные сообщения: 0

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_slizzing.h"
3
4  void check_slizzing(char* string){
5      int i;
6      for(i=0; i<200; 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_slizzing.h>
3  #include <stdlib.h>
4  #include<stdio.h>
5
6  void ui_check_slizzing(){
7      int i;
8      char* string = (char*) malloc(200);
9      FILE * textfile = fopen("somefile", "r");
10     for (i=0; i<3; i++)
11     fgets(string,199, textfile);
12     fclose(textfile);
13     check_slizzing(string);
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

- LimitedStorage(int num1 = 2, int num2 = 7)
  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 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 Qt 5.5.0 MinGW 32bit, операционная система Linux version 3.16.0-4-586.

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

5.1.5 Тестовый план и результаты тестирования

Для статического анализа использовалась утилита Cppcheck.

Результат проверки limitedStorage.c
Статистика:
Ошибки: 0
Предупреждения: 0
Стилистические предупреждения: 0
Предупреждения переносимости: 0

Предупреждения производительности: 0
Информационные сообщения: 0

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 num1 = 2, int num2 = 7);
    void Copy(RationalNum);
    void sum(int);
    void Multi(int);
    void divide(int);
    double ToDouble();
private:

};
class DevNull:public exception{
public:


};

#endif // RATIONALNUM_H
```

rationalnum.cpp

```
#include "rationalnum.h"


RationalNum::RationalNum(int num1, int num2)
```

```
5  {
6      numerator = num1;
7      denominator = num2;
8  }
9  void RationalNum::Copy(RationalNum numb){
10     numerator = numb.numerator;
11     denominator = numb.denominator;
12 }
13 void RationalNum::sum(int num){
14     numerator += num*denominator;
15
16 }
17 void RationalNum::Multi(int Num){
18     numerator *= Num;
19
20 }
21 void RationalNum::divide(int Num){
22     if (Num == 0){
23         DevNull error;
24         throw error;
25     }
26     denominator /= Num;
27
28 }
29 double RationalNum::ToDouble(){
30     return((double)numerator / double(denominator));
31 }
```

Глава 6

Приложение

../sources/subdirproject/cpplib/limitedStorage.cpp

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(double rad = 0.5);
    void convert();
};

#endif // TRANSLATION_H
```

   translation.cpp

```
#include "translation.h"

Translation::Translation(double rad)
```

```
{
 radian = rad;

}

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);
}
```

2. Класс tell_ me_ age манипуляции с возрастом.

tell_ me_ age.h

```
#ifndef TELL_ME_AGE_H
#define TELL_ME_AGE_H


class tell_me_age
{
private:
    int age;
public:
    tell_me_age(int input = 12);
    int checking_age();
    void output();
};

#endif // TELL_ME_AGE_H
```

tell_ me_ age.cpp

```
#include "tell_me_age.h"

tell_me_age::tell_me_age(int input)
{
    age = input;
}

int tell_me_age::checking_age()
{
    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  }
```

3. Класс conversation дублирование четных цифр

conversation.h

```
1  #ifndef CONVERSATION_H
2  #define CONVERSATION_H
3  #include<cmath>
4
5  class conversation
6  {
7  private:
8      int input_num;
9      int output_num;
10 public:
11     conversation(int num = 1234);
12     int convert();
13 };
14
15 #endif // CONVERSATION_H
```

conversation.cpp

```
1  #include "conversation.h"
2
3  conversation::conversation(int num)
4  {
5      input_num = num;
6  }
7
8  int conversation::convert()
9  {
10     int tmp, amount_of_numerals;
11     tmp = input_num;
12
13     for (amount_of_numerals=0; tmp > 0;
           amount_of_numerals++){
14         tmp = tmp / 10;
15     }
```

```
16
17      while(input_num > 0){
18          tmp = input_num / pow(10, amount_of_numerals-1);
19          int remaining_power = pow(10, amount_of_numerals
                -1);
20
21          if (tmp % 2 == 0)
22              output_num = output_num * 100 + tmp * 10 +
                    tmp;
23          else
24              output_num = output_num * 10 + tmp;
25
26          input_num = input_num % remaining_power;
27          amount_of_numerals--;
28      }
29      return output_num;
30  }
```

4. Класс Massive_ without_ nulls для удаления из массива нулей.

massive_ without_ nulls.h

```
1  #ifndef MATRIX_WITHOUT_NULLS_H
2  #define MATRIX_WITHOUT_NULLS_H
3
4
5  class matrix_without_nulls
6  {
7  private:
8      int* array;
9      int* new_array;
10 public:
11     matrix_without_nulls();
12     void converting_matrix();
13 };
14
15 #endif // MATRIX_WITHOUT_NULLS_H
```

matrix_ without_ nulls.cpp

```
1  #include "matrix_without_nulls.h"
2
3  matrix_without_nulls::matrix_without_nulls()
4  {
5      array = new int[20];
6  }
7
8  void matrix_without_nulls::converting_matrix()
9  {
```

```
10      int k, i;
11      for (i=0; i<20; i++){
12
13          for (k=1; k<20-i; k++){
14
15              if (array[i] == 0){
16
17                  array[i] = array[i+k];
18                  array[i+k] = 0;
19
20              }
21              else break;
22          }
23      }
24      for (i = 0; i < 20;i++)
25
26          if (array[i] == 0) {k = i;
27              break;
28          }
29
30  new_array = new int[k];
31      for (i=0; i<k; i++)
32
33          new_array[i] = array[i];
34  delete [] new_array;
35
36  }
```

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  private:
9      string text;
10 public:
11     check_sizzling(string sometext = "restart nimfa vyrer
           ");
12     void find_symbol();
13     int check_symbol(char);
14
15 };
16
```

```
17 #endif // CHECK_SIZZLING_H
```

check_ sizzling.cpp

```
1  #include "check_sizzling.h"
2
3  check_sizzling::check_sizzling(string sometext)
4  {
5      text = sometext;
6
7  }
8
9  void check_sizzling::find_symbol()
10 {
11             int i;
12         for(i=0; i<200; i++)
13             if (text[i] == 'r' || text[i] == 'n'||text[i]
                   == 'v')
14                 switch(check_symbol(text[i+1])){
15                 case 0:
16                     text[i+1] = 'a';
17                     break;
18                 case 1:
19                     text[i+1] = 'u';
20                     break;
21                 case 2:
22                     text[i+1] = 'e';
23                     break;
24                 default: break;
25
26                 }
27 }
28
29 int check_sizzling::check_symbol(char vowel)
30 {
31     if(vowel == 'e') return 0;
32     if(vowel == 'i') return 1;
33     if(vowel == 'y') return 2;
34     return 3;
35 }
```

6.2 Автоматические тесты

```
1 #include <QString>
2 #include <QtTest>
3 #include "check_sizzling.h"
4
```

```
5  class CpptestTest : public QObject
6  {
7      Q_OBJECT
8
9  public:
10     CpptestTest();
11
12 private Q_SLOTS:
13     void testCase1();
14 };
15
16 CpptestTest::CpptestTest()
17 {
18 }
19
20 void CpptestTest::testCase1()
21 {
22     QVERIFY2(true, "Failure");
23 }
24
25 QTEST_APPLESS_MAIN(CpptestTest)
26
27 #include "tst_cpptesttest.moc"
```

```
1  #include <QString>
2  #include <QtTest>
3  #include "conversation.h"
4  #include "translation.h"
5
6  class IlinTest : public QObject
7  {
8      Q_OBJECT
9
10 public:
11     IlinTest();
12
13 private Q_SLOTS:
14    void test_conversation();
15    void test_translation();
16 };
17
18 IlinTest::IlinTest()
19 {
20 }
21
22 void IlinTest::test_conversation()
23 {    int number = 12234;
24         QCOMPARE(conversation(number), 12222344);
25
```

```
26 }
27 void IlinTest::test_translation()
28 {
29     double radians = 0.5;
30     Angle angle;
31     translation(radians, &angle);
32     QCOMPARE(angle.degree, 28);
33     QCOMPARE(angle.min, 39);
34     QCOMPARE(angle.sec, 44);
35
36 }
37 //void IlinTest::test_tell_me_age()
38 //{
39
40 //     // QVERIFY2(true, "Failure");
41 //}
42
43 QTEST_APPLESS_MAIN(IlinTest)
44
45 #include "tst_ilintest.moc"
```