

# 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:

$$1\text{radian} = 180/\pi\text{degrees}$$

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

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, соответствующие градусам, минутам и секундам.

В ходе проектирования было решено выделить одну функцию:

- `void translation(double, Angle*)`

Функция вычисляет переводит из радиан в градусы. Параметрами функции являются переменная типа double и указатель на созданную структуру. Первое значение соответствует числу радиан, переданному пользователем, а в структуру на которую получен указатель 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 автоматическое 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 в таблице:

Входные данные	Ожидаемый результат		Полученный результат	
	Возвращаемое значение	Корни	Возвращаемое значение	Корни
3 5 10	0		0	
23 11 1	1	-0.35620 -0.12206	1	-0.35620 -0.12206
10 0 2	0		0	
23 11 1	1	-1.16031 1	0	-1.16031 0

### 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 выполнения 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 библиотеки из заголовочного файла `stdio.h` для ввода и вывода информации.

### 1.2.3 Проектирование

В ходе проектирования были выделены следующая функция:

- `int tell-me-age(int)`

Функция проверяет число на несколько условий и возвращает один из идентификаторов. Перечисляемый 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 идентифиф  
Статистика:  
Ошибки: 0  
Предупреждения: 0

Стилистические предупреждения: 0  
 Предупреждения переносимости: 0  
 Предупреждения производительности: 0  
 Информационные сообщения: 0

В ходе автоматического тестирования функции `conversation` подавалось число, содержащее четные цифры и проверялось с устно подсчитанным результатом.

Входные данные	Результат	Ожидаемый результат
11 0	12	12
21 10	2	2
19 9	5	5
35 25	0	0
12 0	1	1
31 9	10	10
21 12	10	10
43 48	31	31
2 0	3	3
7 40	19	19
22 0	22	22
26 23	49	49

## 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

```
1 #include "ui_tell_me_age.h"
2 #include "tell_me_age.h"
3
4 void ui_tell_me_age(){
5     int age;
6     printf("Input your age.\n");
7     scanf("%d", &age);
8     int ans = tell_me_age(age);
9     switch(ans){
10    case 0: printf("Вам %d лет!", age); break;
11    case 1: printf("Вам %d года!", age); break;
12    case 2: printf("Вам %d год!", age); break;
13    }
14 }
```

# 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 выполнения 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 библиотеки из заголовочного файла stdio.h для ввода и вывода информации, math.h для выполнения вычислений.

#### 2.1.3 Проектирование

В ходе проектирования были выделены следующая 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 несколько раз вызывалась функция factorization, которой подавались различные числа для факторизации. Затем полученные данные сравнивались с ожидаемым результатом.

Входные данные	Результат	Ожидаемый результат
75	3 5 5	3 5 5
13	13	13
34527	3 17 677	3 17 677

### 2.1.6 Выводы

В ходе выполнения я отработал навыки работы с циклами.

## Листинги

conversation.c

```
1 #include "conversation.h"
2
3
4
5 int conversation(int input_num){
6
7     int Output_num = 0, tmp, amount_of_numerals;
8     tmp = input_num;
9
10    for (amount_of_numerals=0; tmp > 0; amount_of_numerals++)
11    {
12        tmp = tmp / 10;
13    }
14
15    while(input_num > 0){
16        tmp = input_num / pow(10, amount_of_numerals-1);
17        int remaining_power = pow(10, amount_of_numerals-1);
18
19        if (tmp % 2 == 0)
20            Output_num = Output_num * 100 + tmp * 10 + tmp;
21        else
22            Output_num = Output_num * 10 + tmp;
23
24        input_num = input_num % remaining_power;
25        amount_of_numerals--;
26    }
27    return Output_num;
```

ui-conversation.c

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

```

13         array[i+k] = 0;
14
15     }
16     else break;
17 }
18 }
19 printf("\n");
20 for (i = 0; i < 20;i++){
21
22     if (array[i] == 0) {k = i;
23         break;
24     }
25
26 }
27 return k;
28 }

```

#### ui-matrix.c

```

1 #include "ui_matrix.h"
2 #include "matrix.h"
3 #include <stdlib.h>
4 void ui_matrix(){
5     int* array = (int*) malloc(sizeof(int)*20) ;
6     int i;
7
8     FILE *myfile = fopen("myfile.txt", "r");
9     for(i = 0; i< 20; i++)
10         fscanf(myfile, "%d", &array[i]);
11
12     fclose(myfile);
13
14     int k = matrix_not_zero(array);
15
16     FILE *Output = fopen("Output.txt", "w");
17     int* new_array = (int*) malloc(sizeof(int)*20);
18     for (i=0; i<k; i++){
19
20         new_array[i] = array[i];
21         fprintf(Output,"%d ", new_array[i]);
22
23     }
24     fclose(Output);
25     free(array);
26     free(new_array);
27     printf("Done.\n");
28 }

```

# 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 библиотeki.

#### 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 создан класс 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 автоматическое 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

В ходе автоматического тестирования вызывалась функция `splitArray`, которой подавался массив с целыми числами. Затем полученные в матрице данные сравнивались с ожидаемым результатом.

### 5.1.6 Выводы

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

### Листинги

rationalnum.h

```
1 #ifndef RATIONALNUM_H
2 #define RATIONALNUM_H
3 #include <iostream>
4 #include <exception>
5 using namespace std;
6
7 class RationalNum
8 {
9     int numerator;
10    int denominator;
11 public:
12    RationalNum(int num1 = 2, int num2 = 7);
13    void Copy(RationalNum);
14    void sum(int);
15    void Multi(int);
16    void divide(int);
17    double ToDouble();
18 private:
19
20 };
21 class DevNull:public exception{
22 public:
23
24
25 };
26
27 #endif // RATIONALNUM_H
```

rationalnum.cpp

```

1 #include "rationalnum.h"
2
3
4 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 QuadEquation, квадратное уравнение.

QuadEquationClass.h

QuadEquationClass.cpp

2. Класс Date для работы с датой и временем.

date.h

date.cpp

3. Класс Factorizator для разложения числа на простые множители.

Factorizator.h

Factorizator.cpp

4. Класс ArraySplitter для разбиения массива.

5. Класс Drug, медикамент.

### 6.2 Классы исключений

### 6.3 Автоматические тесты