Программирование на C и C++

В. В. Дьячков

23 декабря 2015 г.

Оглавление

Глава 1

Основные конструкции языка

1.1 Задание 1.1. Перевод дюймов в метрическую систему

1.1.1 Задание

Перевести длину отрезка из дюймов в метры, сантиметры и миллиметры.

1.1.2 Теоретические сведения

Для реализации данной задачи была использована структура `struct` для удобства представления трех величин: метров, сантиметров и миллиметров.

Так же были использованы стандартные функции ввода-вывода `scanf`, `printf`, `puts` из стандартной библиотеки языка `C`, объявленные в заголовочном файле *stdio.h*.

При помощи операторов ветвления `if-else` и `switch` реализовано интерактивное подменю для более удобного взаимодействия пользователя с программой.

Для решения поставленной задачи воспользовался метричский фактом: 1 дюйм = 2.54 см.

1.1.3 Проектирование

В ходе проектирования было решено выделить 5 функций:

1. Перевод дюймов в метры, сантиметры и миллиметры

   ```
   void calculating_inch_to_cm(int, Meters *);
   ```

Параметрами функции являются целочисленное `int` значение дюймов и указатель на структуру `Meters`, в которой будут содержаться значения для метров, сантиметров, миллиметров в результате работы функции. Структура объявлена в заголовочном файле `inch_to_cm.h`

2. Меню с первоначальным пользовательским взаимодействием

   `void menu_inch_to_cm();`

   Пользователю предлагается выбрать консольный ввод, вызов справки, возварт к главное меню или завершение программы.

3. Основное пользовательское взаимодействие

   `void input_inch_to_cm();`

   Пользователю предлагается ввести дюймы, после чего вызывается функция для перевода в метрическую систему и функция для вывода получившегося результата в консоль.

4. Вывод в консоль получившегося результата

   `void show_inch_to_cm(int, Meters);`

   Параметрами функции являются целочисленное `int` значение дюймов заданное пользователем, и структура `Meters`, в которой содержаться значения для метров, сантиметров, миллиметров в результате работы функции. Структура объявлена в заголовочном файле `inch_to_cm.h`

5. Вспомогательная информация

   `void help_inch_to_cm();`

   Вывод в консоль формулировки задания, помагающая пользователю в использовании программы.

1.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.1, компилятор GCC 4.8.4 (x86 64 bit), операционная система Linux Mint 17.2 Cinnamon 64 bit. В процессе выполнения задания производилось ручное тестирование. Модульное тестирование реализовано при помощи фреймворка QtTest.

1.1.5 Тестовый план и результаты тестирования

В таблице 4.1 представлены значения дюймов использованные при тестировании и ожидаемые значения для метров, сантиметров и миллиметров, а также отметка о результате теста.

Таблица 1.1: Тестовый план и результаты тестирования перевода дюймов в метрическую систему

Дюймы	Метры	Сантиметры	Миллиметры	Тип теста	Результат
301	7	64	5.4	Модульный	Успешно
100	2	54	0	Ручной	Успешно

Все тесты пройдены успешно. Листинги модульных тестов приведены в приложении 5.1.7.

1.1.6 Выводы

При выполнении задания были закреплены навыки в работе с основными конструкциями языка `C` и получен опыт в организации многофайлового проекта и создании модульных тестов.

Данная задача также была переписана на языке `C++` с использованием объектно-ориентироанного проектироания. Основой созданного класса `Meter` стала одноименная струтура из этой главы. Листинги данного класса приведены в приложении 5.1.7.

1.1.7 Листинги

inch_to_cm.h

```h
#ifndef INCH_TO_CM
#define INCH_TO_CM

#ifdef __cplusplus
extern "C" {
#endif

typedef struct
{
    int m;
    int cm;
    double mm;
} Meters;

void calculating_inch_to_cm(int, Meters *);

#ifdef __cplusplus
}
#endif

#endif // INCH_TO_CM
```

inch_to_cm.c

```c
#include "inch_to_cm.h"

void calculating_inch_to_cm(int inches, Meters * meter)
{
    double total_mm = inches * 25.4;

    meter->m = total_mm / 1000;
    total_mm -= meter->m * 1000;

    meter->cm = total_mm / 10;
    total_mm -= meter->cm * 10;

    meter->mm = total_mm;
}
```

ui_inch_to_cm.h

```h
#ifndef UI_INCH_TO_CM
#define UI_INCH_TO_CM

#include "inch_to_cm.h"

void menu_inch_to_cm();
```

```
7  void input_inch_to_cm();
8  void show_inch_to_cm(int, Meters);
9  void help_inch_to_cm();
10
11 #endif // UI_INCH_TO_CM
```

ui_inch_to_cm.c

```
1  #include "ui.h"
2  #include "inch_to_cm.h"
3  #include "ui_inch_to_cm.h"
4
5  void menu_inch_to_cm()
6  {
7      int num;
8      puts("Translate inches to meters:");
9      puts("1. Input inches");
10     puts("2. Help");
11     puts("9. Back to main menu");
12     puts("0. Exit");
13     printf(">>> ");
14     if (scanf("%d", &num) == 1)
15     {
16         switch (num)
17         {
18         case 0:
19             break;
20         case 1:
21             input_inch_to_cm(); menu_inch_to_cm(); break;
22         case 2:
23             help_inch_to_cm(); menu_inch_to_cm(); break;
24         case 9:
25             main_menu(); break;
26         default:
27             puts("Error! Invalid number.\n"); menu_inch_to_cm
                   (); break;
28         }
29     }
30     else
31     {
32         puts("Error! Input a number.\n");
33         __fpurge(stdin);
34         menu_inch_to_cm();
35     }
36 }
37
38 void input_inch_to_cm()
39 {
40     int inches;
41     Meters meter;
```

```
42      printf("Input inches: ");
43      scanf("%d", &inches);
44      calculating_inch_to_cm(inches, &meter);
45      show_inch_to_cm(inches, meter);
46      printf("\n");
47  }
48
49  void show_inch_to_cm(int inches, Meters meter)
50  {
51      printf("%d inches = %d m %d cm %.1f mm\n", inches, meter.
            m, meter.cm, meter.mm);
52  }
53
54  void help_inch_to_cm()
55  {
56      puts("HELP: Перевести длину отрезка из дюймов в метры, са
            нтиметры и миллиметры.");
57  }
```

1.2 Задание 1.2. Расчет времени

1.2.1 Задание

Определить, за какое время путник одолел первую половину пути, двигаясь T1 часов со скоростью V1, T2 часов со скоростью V2, T3 часов со скоростью V3.

1.2.2 Теоритические сведения

Для того, чтобы абстрагироваться от количества частей пути, на которых путник двигался с различной скоростью, в реализации задачи был использован макрос `#define NUMBER_OF_PIECES 3`. Благодаря такому приему с помощью лишь одной замены в заголовочном файле *time.h* мы можем изменить количество таких участков, не потеряв работоспособность программы.

Так же были использованы стандартные функции ввода-вывода `scanf`, `printf`, `puts` из стандартной библиотеки языка `C`, объявленные в заголовочном файле *stdio.h*.

При помощи операторов ветвления `if-else` и `switch` реализовано интерактивное подменю для более удобного взаимодействия пользователя с программой. С использованием оператора цикла `for` происходит иттерирование по каждому участку пути.

Для решения поставленной задачи были использованы следующие математические факты:

- чтобы найти путь на отдельном участке пути необхадимо умножить скорость на даном участке на время, затраченное на прохождение этого участка;
- чтобы найти общий путь необходимо сложить пути всех участков.

1.2.3 Проектирование

В ходе проектирования было решено выделить 6 функций:

1. Нахождение половины пройденного пути

   `double halfdistance_time(double *, double *);`

   В качестве передаваемых параметров используются 2 указателя на тип `double *` – **скорости** на участках пути и **время**, затраченное

на прохождение каждого такого участка. Возвращаемым значением является расстояние типа `double`, равное половине пройденного пути.

2. Вычисление времени, затраченного на прохождение первой половины пути

   ```
   double calculating_time(double *, double *);
   ```

   В качестве передаваемых параметров используются 2 указателя на тип `double *` – **скорости** на участках пути и **время**, затраченное на прохождение каждого такого учатска. Возвращаемым значением является рассчитанное время типа `double`, затраченное на прохождение половины пути.

3. Меню с начальным пользовательским взаимодействием

   ```
   void menu_time();
   ```

   Пользователю предлагается выбрать консольный ввод, вызов справки, возварт к главное меню или завершение программы.

4. Основное пользовательское взаимодействие

   ```
   void input_time();
   ```

   Пользователю предлагается последовательно ввести скорость и время для каждого участка пути, после чего вызывается функция для вычисления времени, затраченного на половину пути и функция для вывода получившегося результата в консоль.

5. Вывод в консоль получившегося результата

   ```
   void show_time(double);
   ```

   Для этого в качестве параметров передаются вещественное число `double` – вычситанное значение времени.

6. Вспомогательная информация

   ```
   void help_time();
   ```

   Вывод в консоль формулировки задания, помагающая пользователю в использовании программы.

1.2.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.1, компилятор GCC 4.8.4 (x86 64 bit), операционная система Linux Mint 17.2 Cinnamon 64 bit. В процессе выполнения задания производилось ручное тестирование. Модульное тестирование реализовано при помощи фреймворка QtTest.

1.2.5 Тестовый план и результаты тестирования

В таблице 1.2 представлены значения дюймов использованные при тестировании и ожидаемые значения для метров, сантиметров и миллиметров, а также отметка о результате теста.

Таблица 1.2: Тестовый план и результаты тестирования расчета времени, затраченнего на половину пути

V1	V2	V3	T1	T2	T3	Результат	Тип теста	Результат
60	80	100	4	2	5	6.5	Модульный	Успешно
50	100	150	1	1	1	2.0	Ручной	Успешно

Все тесты пройдены успешно. Листинги модульных тестов приведены в приложении 5.1.7.

1.2.6 Выводы

При выполнении задания закреплены навыки в работе с основными конструкциями языка `C` и получен опыт в организации многофайлового проекта и создании модульных тестов.

Данная задача также была переписана на языке `C++` с использованием объектно-ориентироанного проектироания. Основой созданного класса `Route` стали скорости на участках пути и время, за которое данные участки преодолевались. Листинги данного класса приведены в приложении 5.1.7.

1.2.7 Листинги

time.h

```
#ifndef TIME
#define TIME

#ifdef __cplusplus
extern "C" {
#endif

#define NUMBER_OF_PIECES 3
double halfdistance_time(double *, double *);
double calculating_time(double *, double *);

#ifdef __cplusplus
}
#endif

#endif // TIME
```

time.c

```
#include "time.h"

double halfdistance_time(double * velocity, double * time)
{
    double s = 0;
    int i;
    for (i = 0; i < NUMBER_OF_PIECES ; i++)
        s += velocity[i]*time[i];
    return s/2;
}

double calculating_time(double * velocity, double * time)
{
    double halfdist = halfdistance_time(velocity, time);
    double total_time = 0;
    int i;
    for (i = 0; i < NUMBER_OF_PIECES; i++)
    {
        if (velocity[i]*time[i]<halfdist)
        {
            total_time += time[i];
            halfdist = halfdist - velocity[i]*time[i];
        }
        else
        {
            total_time += halfdist/velocity[i];
            halfdist = 0;
```

```
        }
    }
    return total_time;
}
```

ui_time.h

```
#ifndef UI_TIME
#define UI_TIME

#include "time.h"

void menu_time();
void help_time();
void input_time();
void show_time(double);

#endif // UI_TIME
```

ui_time.c

```
#include "ui.h"
#include "time.h"
#include "ui_time.h"

void menu_time()
{
    int num;
    puts("Calculation time of half way:");
    puts("1. Input velocity and time");
    puts("2. Help");
    puts("9. Back to main menu");
    puts("0. Exit");
    printf(">>> ");
    if (scanf("%d", &num) == 1)
    {
        switch (num)
        {
        case 0:
            break;
        case 1:
            input_time(); menu_time(); break;
        case 2:
            help_time(); menu_time(); break;
        case 9:
            main_menu(); break;
        default:
            puts("Error! Invalid number.\n"); menu_time();
               break;
        }
```

```
29      }
30      else
31      {
32          puts("Error! Input a number.\n");
33          __fpurge(stdin);
34          main_menu();
35      }
36  }
37
38  void input_time()
39  {
40      double velocity[NUMBER_OF_PIECES];
41      double time[NUMBER_OF_PIECES];
42      puts("Input velocity and time:");
43      int i;
44      for (i = 0; i < NUMBER_OF_PIECES ; i++)
45      {
46          printf("V[%d] = ", i+1);
47          scanf("%lf", &velocity[i]);
48          printf("T[%d] = ", i+1);
49          scanf("%lf", &time[i]);
50      }
51      double total_time = calculating_time(velocity, time);
52      show_time(total_time);
53      printf("\n");
54  }
55
56  void show_time(double time)
57  {
58      printf("Required time is %.2f hours.\n", time);
59  }
60
61  void help_time()
62  {
63      puts("HELP: Определить, за какое время путник одолел перв
           ую половину пути, двигаясь T1 часов со скоростью V1,
           T2 часов со скоростью V2, T3 часов со скоростью V3.");
64  }
```

Глава 2

Циклы

2.1 Задание 2. Палиндром

2.1.1 Задание

Проверить, является ли заданное число палиндромом.

2.1.2 Теоретические сведения

Для реализации данной задачи были использованы стандартные функции ввода-вывода `scanf`, `printf`, `puts` из стандартной библиотеки языка `C`, объявленные в заголовочном файле *stdio.h*.

При помощи операторов ветвления `if-else` и `switch` реализовано интерактивное подменю для более удобного взаимодействия пользователя с программой.

Для решения поставленной задачи использовался факт: число называется палиндромом, если первая цифра равна последней, вторая предпоследней и так далее.

2.1.3 Проектирование

В ходе проектирования было решено выделить 5 функций:1

1. Проверка числа на палиндром

   ```
   int is_palindrome(char *);
   ```

   Параметром функции является указатель на строку `char *`, содержащую число, представленное в виде символов `char`. Возвращаемое значение `int` равно `1`, если число является палиндромом и `0`, если оно таковым не является.

2. Меню с первоначальным пользовательским взаимодействием

   `void menu_palindrome();`

   Пользователю предлагается выбрать консольный ввод, вызов справки, возварт к главное меню или завершение программы.

3. Основное пользовательское взаимодействие

   `void input_palindrome();`

   Пользователю предлагается ввести число, после чего вызывается функция для опредления палиндрома и функция для вывода получившегося результата в консоль.

4. Вывод в консоль получившегося результата

   `void show_palindrome(char *, int);`

   Параметрами функции являеются указатель на строку `char *` – число, которое проверяется на палиндром, и целочисленное значение `int` равное `1`, если число является палиндромом и `0`, если оно таковым не является.

5. Вспомогательная информация

   `void help_palindrome();`

   Вывод в консоль формулировки задания, помагающая пользователю в использовании программы.

2.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.1, компилятор GCC 4.8.4 (x86 64 bit), операционная система Linux Mint 17.2 Cinnamon 64 bit. В процессе выполнения задания производилось ручное тестирование. Модульное тестирование реализовано при помощи фреймворка QtTest.

2.1.5 Тестовый план и результаты тестирования

В таблице 4.1 представлены значения дюймов использованные при тестировании и ожидаемые значения для метров, сантиметров и миллиметров, а также отметка о результате теста.

Таблица 2.1: Тестовый план и результаты тестирования перевода дюймов в метрическую систему

Число	Палиндром	Тип теста	Результат
2311441132	Да	Модульный	Успешно
525321	Нет	Модульный	Успешно
123464321	Да	Ручной	Успешно
165332	Нет	Ручной	Успешно

Все тесты пройдены успешно. Листинги модульных тестов приведены в приложении 5.1.7.

2.1.6 Выводы

При выполнении задания были закреплены навыки в работе с оператором цикла `for` и получен опыт в организации многофайлового проекта и создании модульных тестов.

Данная задача также была переписана на языке `C++` с использованием объектно-ориентироанного проектироания. Основой созданного класса `Number` стало число с вызываемым для него методом, определяющим, является ли это число палиндромом. Листинги данного класса приведены в приложении 5.1.7.

2.1.7 Листинги

palindrome.h

```
#ifndef PALINDROME
#define PALINDROME

#ifdef __cplusplus
extern "C" {
#endif

#include "string.h"

int is_palindrome(char *);

#ifdef __cplusplus
}
#endif

#endif // PALINDROME
```

palindrome.c

```
#include "palindrome.h"

int is_palindrome(char * number)
{
    int len = strlen(number);
    int halflen = len / 2;

    int i = 0;
    for (i = 0; i < halflen; i++)
        if (number[i] != number[len - i - 1])
            return 0;
    return 1;
}
```

ui_palindrome.h

```
#ifndef UI_PALINDROME
#define UI_PALINDROME

#include "palindrome.h"

void menu_palindrome();
void input_palindrome();
void show_palindrome(char *, int);
void help_palindrome();

#endif // UI_PALINDROME
```

ui_palindrome.c

```c
#include "ui.h"
#include "palindrome.h"
#include "ui_palindrome.h"

void menu_palindrome()
{
    int num;
    puts("Definition of palindrome:");
    puts("1. Input a number");
    puts("2. Help");
    puts("9. Back to main menu");
    puts("0. Exit");
    printf(">>> ");
    if (scanf("%d", &num) == 1)
    {
        switch (num)
        {
        case 0:
            break;
        case 1:
            input_palindrome(); menu_palindrome(); break;
        case 2:
            help_palindrome(); menu_palindrome(); break;
        case 9:
            main_menu(); break;
        default:
            puts("Error! Invalid number.\n"); menu_palindrome
                (); break;
        }
    }
    else
    {
        puts("Error! Input a number.\n");
        __fpurge(stdin);
        menu_palindrome();
    }
}

void input_palindrome()
{
    char number[20];
    puts("Input a number:");
    scanf("%s", number);
    int is_palin = is_palindrome(number);
    show_palindrome(number, is_palin);
    printf("\n");
}
```

```
47
48 void show_palindrome(char * number, int is_palin)
49 {
50     if (is_palin)
51         printf("%s is palindrome\n", number);
52     else
53         printf("%s isn't palindrome\n", number);
54 }
55
56 void help_palindrome()
57 {
58     puts("HELP: Проверить, является ли заданное число NUMBER
        палиндромом (симметричным, первая цифра равна последне
        й и так далее).");
59 }
```

Глава 3

Массивы

3.1 Задание 3. Игра

3.1.1 Задание

По кругу располагаются n человек. Ведущий считает по кругу, начиная с первого, и выводит m-ого человека. Круг смыкается, счет возобновляется со следующего; так продолжается, пока в круге не останется только один человек. Найти номер этого человека.

3.1.2 Теоретические сведения

Для реализации данной задачи были использованы ряд стандартных функций ввода-вывода, таких как `scanf`, `printf`, `puts`, `fopen`, `fclose`, `fscanf`, `fprintf` из стандартной библиотеки языка `C`, объявленные в заголовочном файле *stdio.h*, а так же функции для работы с памятью `malloc` и `free`, объявленные в заголовочном файле *stdlib.h*.

При помощи операторов ветвления `if-else` и `switch` реализовано интерактивное подменю для более удобного взаимодествия пользователя с программой. При помощи цикла `for` и `while` происходит обращение к эллементам массива.

Для решения поставленной задачи использовался разработанный алгоритм для определения искомого номера, основанный на иттериовании по массиву.

3.1.3 Проектирование

В ходе проектирования было решено выделить 6 функций:

1. Определение номера победителя

   `int determine_the_winner(int, int);`

   Параметром функции является два целочисленных значения `int` – количество игроков и номер, игрока с которым выводят из круга. Возвращаемое значение `int` равно номеру победившего игрока.

2. Меню с первоначальным пользовательским взаимодействием

   `void menu_circle_game();`

   Пользователю предлагается выбрать консольный ввод, файловый ввод, вызов справки, возварт к главное меню или завершение программы.

3. Консольное взаимодействие с пользователем

   `void input_console_circle_game();`

   Пользователю предлагается ввести число, после чего вызывается функция для опредления победителя и функция для вывода получившегося результата в консоль.

4. Файловое взаимодействие с пользователем

   `void input_file_circle_game();`

   Пользователю предлагается ввести имя файла, откуда программа должна получить исходные данные, и имя файла, куда будет происходить печать результата, после чего вызывается функция для опредления победителя и вывод результата в указанный файл.

5. Вывод в консоль получившегося результата

   `void show_circle_game(int);`

   Параметром функции является и целочисленное значение `int` равное номеру победившего игрока.

6. Вспомогательная информация

   `void help_circle_game();`

   Вывод в консоль формулировки задания, помагающая пользователю в использовании программы.

3.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.1, компилятор GCC 4.8.4 (x86 64 bit), операционная система Linux Mint 17.2 Cinnamon 64 bit. В процессе выполнения задания производилось ручное тестирование. Модульное тестирование реализовано при помощи фреймворка QtTest.

3.1.5 Тестовый план и результаты тестирования

В таблице 4.1 представлены значения дюймов использованные при тестировании и ожидаемые значения для метров, сантиметров и миллиметров, а также отметка о результате теста.

Таблица 3.1: Тестовый план и результаты тестирования перевода дюймов в метрическую систему

Игроков	Выводят номер	Победитель	Тип теста	Результат
32	5	13	Модульный	Успешно
9	45	7	Модульный	Успешно
5	3	4	Ручной	Успешно

Все тесты пройдены успешно. Листинги модульных тестов приведены в приложении 5.1.7.

3.1.6 Выводы

При выполнении задания были закреплены навыки в работе с массивами и получен опыт в организации многофайлового проекта и создании модульных тестов.

Данная задача также была переписана на языке `C++` с использованием объектно-ориентироанного проектироания. Был создан класс `CircleGame`. Листинги данного класса приведены в приложении 5.1.7.

3.1.7 Листинги

circle_game.h

```
#ifndef CIRCLE_GAME
#define CIRCLE_GAME

#include "stdlib.h"

#ifdef __cplusplus
extern "C" {
#endif

int determine_the_winner(int, int);

#ifdef __cplusplus
}
#endif

#endif // CIRCLE_GAME
```

circle_game.c

```
#include "circle_game.h"

int determine_the_winner(int number_of_players, int
    number_for_kick)
{
    int * is_player_in_game;
    is_player_in_game = (int *) malloc(sizeof(int) *
        number_of_players);
    int i;
    for (i = 0; i < number_of_players; i++)
    {
        is_player_in_game[i] = 1;
    }
    int j, cursor = 0;

    for (i = 0; i < number_of_players-1; i++) // проиграть до
        лжны number_of_players-1 игроков
    {
        for (j = 0; j < number_for_kick-1; cursor++) // счита
            ем подряд number_for_kick-1 игроков, остающихся в
            игре
            if (is_player_in_game[cursor % number_of_players
                ])
                j++;

        while (is_player_in_game[cursor % number_of_players]
            == 0) // ищем следующего number_for_kick игрока, к
```

```
            оторый пока в игре
21              cursor++;
22
23          is_player_in_game[cursor % number_of_players] = 0; //
                number_for_kick игрок объявляется проигравшим
24          cursor += 1;
25      }
26      for(i = 0; i < number_of_players && is_player_in_game[i
            ]==0; i++);
27      free(is_player_in_game);
28      return i + 1;
29  }
```

ui_circle_game.h

```
1  #ifndef UI_CIRCLE_GAME
2  #define UI_CIRCLE_GAME
3
4  #include "circle_game.h"
5
6  void menu_circle_game();
7  void input_console_circle_game();
8  void input_file_circle_game();
9  void show_circle_game(int);
10 void help_circle_game();
11
12 int __fpurge(FILE *stream);
13
14 #endif // UI_CIRCLE_GAME
```

ui_circle_game.c

```
1  #include "ui.h"
2  #include "circle_game.h"
3  #include "ui_circle_game.h"
4
5  void menu_circle_game()
6  {
7      int num;
8      puts("Circle game:");
9      puts("1. Input numbers from console");
10     puts("2. Input numbers from files");
11     puts("3. Help");
12     puts("9. Back to main menu");
13     puts("0. Exit");
14     printf(">>> ");
15     if (scanf("%d", &num) == 1)
16     {
17         switch (num)
18         {
```

```
19          case 0:
20              break;
21          case 1:
22              input_console_circle_game(); menu_circle_game();
                    break;
23          case 2:
24              input_file_circle_game(); menu_circle_game();
                    break;
25          case 3:
26              help_circle_game(); menu_circle_game(); break;
27          case 9:
28              main_menu(); break;
29          default:
30              puts("Error! Invalid number.\n");
                    menu_circle_game(); break;
31          }
32      }
33      else
34      {
35          puts("Error! Input a number.\n");
36          __fpurge(stdin);
37          menu_circle_game();
38      }
39  }
40
41  void input_console_circle_game()
42  {
43      int n, m;
44      printf("Input the number of players: ");
45      scanf("%d", &n);
46      printf("Input the parameter selection: ");
47      scanf("%d", &m);
48      int winner = determine_the_winner(n, m);
49      show_circle_game(winner);
50      printf("\n");
51  }
52
53  void input_file_circle_game()
54  {
55      int n, m;
56      printf("Write the input file (0 for \"input\"): ");
57      char i_filename[20];
58      scanf("%s", i_filename);
59      if (i_filename[0] == '0')
60          strcpy(i_filename, "input");
61      FILE * read;
62
63      if ((read = fopen(i_filename, "r")) == NULL)
64      {
```

```
65          puts("Error! Cannot open input file. Input
                interrupted.\n");
66          menu_circle_game();
67      }
68      else
69      {
70          fscanf(read, "%d %d", &n, &m);
71          fclose(read);
72
73          printf("Write the output file (0 for \"output\"): ");
74          char o_filename[20];
75          scanf("%s", o_filename);
76          if (o_filename[0] == '0')
77              strcpy(o_filename, "output");
78
79          FILE * write = fopen(o_filename, "w");
80          if (write == NULL)
81          {
82              puts("Error! Cannot open output file. Input
                    interrupted.\n");
83              menu_circle_game();
84          }
85          else
86          {
87              int winner = determine_the_winner(n, m);
88              fprintf(write, "#%d won this game", winner);
89              fclose(write);
90              printf("Check result in %s\n\n", o_filename);
91          }
92      }
93  }
94
95  void show_circle_game(int winner)
96  {
97      printf("#%d won this game!\n", winner);
98  }
99
100 void help_circle_game()
101 {
102     puts("HELP: По кругу располагаются NUM_OF_PLAYERS человек
        . Ведущий считает по кругу, начиная с первого, и вывод
        ит NUM_FOR_KICK-го человека. Круг смыкается, счет возо
        бновляется со следующего; так  продолжается, пока в кр
        уге не останется только один человек. Найти номер этог
        о человека.");
103 }
```

Глава 4

Строки

4.1 Задание 4. Фразы

4.1.1 Задание

Проверить, что все фразы в тексте начинаются с прописной буквы и при необходимости откорректировать текст.

4.1.2 Теоретические сведения

Для реализации данной задачи были использованы ряд стандартных функций ввода-вывода, таких как `scanf`, `printf`, `puts`, `fopen`, `fclose`, `fscanf`, `fprintf` из стандартной библиотеки языка `C`, объявленные в заголовочном файле *stdio.h*, а так же функция для преобразования строчных букв в прописные `toupper`, объявленная в заголовочном файле *ctype.h* и функция `strcpy`, объявленная в библиотеке *string.h*.

При помощи операторов ветвления `if-else` и `switch` реализовано интерактивное подменю для более удобного взаимодействия пользователя с программой. При помощи цикла `for` и происходит обращение к символам строки.

Для решения поставленной задачи использовался правило русского языка: новое предложение должно начинаться с прописной буквы.

4.1.3 Проектирование

В ходе проектирования было решено выделить 6 функций:

1. Проверка и исправление переданной строки

   ```
   void upper_case_phrases(char *);
   ```

Параметром функции является указтель на строку `char *`, содержащий адрес переданной строки.

2. Меню с первоначальным пользовательским взаимодействием

   `void menu_phrases();`

   Пользователю предлагается выбрать консольный ввод, файловый ввод, вызов справки, возварт к главное меню или завершение программы.

3. Консольное взаимодействие с пользователем

   `void input_console_phrases();`

   Пользователю предлагается ввести строку, в которой необхадимо проверить правильность напасания фраз, после чего вызывается функция исправляющая ошибки и функция для вывода полученной строки в консоль.

4. Файловое взаимодействие с пользователем

   `void input_file_phrases();`

   Пользователю предлагается ввести имя файла, откуда программа должна получить исходную строку, и имя файла, куда будет происходить печать исправленной строки, после чего вызывается функция для исправления ошибок и вывод результата в указанный файл.

5. Вывод в консоль получившегося результата

   `void show_phrases(char *);`

   Параметром функции является указтель на строку `char *`, содержащий адрес исправленной строки.

6. Вспомогательная информация

   `void help_phrases();`

   Вывод в консоль формулировки задания, помагающая пользователю в использовании программы.

4.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.1, компилятор GCC 4.8.4 (x86 64 bit), операционная система Linux Mint 17.2 Cinnamon 64 bit. В процессе вы-

полнения задания производилось ручное тестирование. Модульное тестирование реализовано при помощи фреймворка QtTest.

4.1.5 Тестовый план и результаты тестирования

В таблице 4.1 представлены значения дюймов использованные при тестировании и ожидаемые значения для метров, сантиметров и миллиметров, а также отметка о результате теста.

Таблица 4.1: Тестовый план и результаты тестирования перевода дюймов в метрическую систему

Исходная строка	Исправленная строка	Тип теста	Результат
check.check check.	Check.Check check.	Модульный	Успешно
One. two three. four.	One. Two three. Four.	Ручной	Успешно

Все тесты пройдены успешно. Листинги модульных тестов приведены в приложении 5.1.7.

4.1.6 Выводы

При выполнении задания были закреплены навыки в работе со строками и получен опыт в организации многофайлового проекта и создании модульных тестов.

Данная задача также была переписана на языке `C++` с использованием объектно-ориентироанного проектироания. Основой созданного класса `Phrases` стала строка и метод, редактирующий эту строку. Листинги данного класса приведены в приложении 5.1.7.

4.1.7 Листинги

phrases.h

```
#ifndef PHRASES_H
#define PHRASES_H

#ifdef __cplusplus
extern "C" {
#endif

#include "stdio.h"
#include "ctype.h"
void upper_case_phrases(char *);

#ifdef __cplusplus
}
#endif

#endif // PHRASES_H
```

phrases.c

```
#include "phrases.h"

void upper_case_phrases(char * str)
{
    int i;
    str[0] = toupper(str[0]);
    for (i = 0; str[i] != '\0'; i++)
    {
        if (str[i] == '.')
        {
            int beginingOfSentence = i+1;
            if (str[i+1] == ' ')
                beginingOfSentence++;
            str[beginingOfSentence] = toupper(str[
               beginingOfSentence]);
        }
    }
}
```

ui_phrases.h

```
#ifndef UI_PHRASES
#define UI_PHRASES

#include "phrases.h"
#include <stdio.h>
#include "string.h"

```

```
8  void menu_phrases();
9  void input_console_phrases();
10 void input_file_phrases();
11 void show_phrases(char *);
12 void help_phrases();
13
14 #endif // UI_PHRASES
```

ui_phrases.c

```
1  #include "ui.h"
2  #include "phrases.h"
3  #include "ui_phrases.h"
4
5  void menu_phrases()
6  {
7      int num;
8      puts("Inspection of phrases:");
9      puts("1. Input text from console");
10     puts("2. Input text from files");
11     puts("3. Help");
12     puts("9. Back to main menu");
13     puts("0. Exit");
14     printf(">>> ");
15     if (scanf("%d", &num) == 1)
16     {
17         switch (num)
18         {
19         case 0:
20             break;
21         case 1:
22             input_console_phrases(); menu_phrases(); break;
23         case 2:
24             input_file_phrases(); menu_phrases(); break;
25         case 3:
26             help_phrases(); menu_phrases(); break;
27         case 9:
28             main_menu(); break;
29         default:
30             puts("Error! Invalid number.\n"); menu_phrases();
                   break;
31         }
32     }
33     else
34     {
35         puts("Error! Input a number.\n");
36         __fpurge(stdin);
37         menu_phrases();
38     }
39 }
```

```
40
41 void input_console_phrases()
42 {
43     char str[100];
44     puts("Input the text: ");
45     scanf("%*c");
46     gets(str);
47     upper_case_phrases(str);
48     show_phrases(str);
49     printf("\n");
50 }
51
52 void input_file_phrases()
53 {
54     char str[100];
55     printf("Write the input file (0 for \"input\"): ");
56     char i_filename[20];
57     scanf("%s", i_filename);
58     if (i_filename[0] == '0')
59         strcpy(i_filename, "input");
60     FILE * read;
61
62     if ((read = fopen(i_filename, "r")) == NULL)
63     {
64         puts("Error! Cannot open input file. Input
               interrupted.\n");
65         menu_circle_game();
66     }
67     else
68     {
69         fgets(str, 99, read);
70         fclose(read);
71
72         printf("Write the output file (0 for \"output\"): ");
73         char o_filename[20];
74         scanf("%s", o_filename);
75         if (o_filename[0] == '0')
76             strcpy(o_filename, "output");
77
78         FILE * write = fopen(o_filename, "w");
79         if (write == NULL)
80         {
81             puts("Error! Cannot open output file. Input
                   interrupted.\n");
82             menu_circle_game();
83         }
84         else
85         {
86             upper_case_phrases(str);
```

```
87              fprintf(write, "%s", str);
88              fclose(write);
89              printf("Check result in %s\n\n", o_filename);
90          }
91      }
92  }
93
94  void show_phrases(char * str)
95  {
96      printf("%s", str);
97  }
98
99  void help_phrases()
100 {
101     puts("HELP: Проверить, что все фразы в тексте  начинаются
            с прописной буквы и при необходимости откорректироват
        ь текст.");
102 }
```

Глава 5

Инкапсуляция

5.1 Задание 5. Массив

5.1.1 Задание

Реализовать класс МАССИВ (целых чисел, переменного размера). Требуемые методы: конструктор, деструктор, копирование, индексация.

5.1.2 Теоретические сведения

Для реализации данной задачи были использованы библиотечные потоки ввода-вывода `STL`, такие как `cin` и `cout`, объявленные в заголовочном файле *iostream*.

При помощи цикла `for` и происходит иттерирование по массиву. С помощью операторов `try`, `throw` и `catch` реализована обработка исключений. Работа с памятью происходит при помощи операторов `new` и `delete`.

Для решения поставленной задачи использовалось представление класса МАССИВ, объявленное в условии.

5.1.3 Проектирование

В ходе проектирования было решено выделить класс Array – для представления сущности массива и класс ArrayApp, занимающийся ввод и выводом массива в консоль.

Класс `Array` было решено спроектировать следующим образом:

1. Поле, содержащее размер массива

   ```
   int size;
   ```

Использовался тип `int`, т.к. число элементов в массиве является целым числом.

2. Поле, содержащее адрес массива

   ```
   int * ptn;
   ```

   Было решено использовать целочисленный массив, поэтому `ptn` содержит адрес массива целых чисел.

3. Конструкторы

   ```
   Array(int size = 10);
   ```

   Пользователь задает размер массива. Если этот размер меньше нуля, то инструкцией `throw` генерируется объект `OutOfRange`. Если же объяект создается без параметров, используется аргумент по умолчанию.

   ```
   Array(Array &);
   ```

   Конструктор копирования инициализирует создаваемый объект с помощью другого объекта этого класса.

4. Метод, возвращающий размер массива

   ```
   int get_size() const;
   ```

   Возращаемым знчением является текущий размер массива. Метод объявлен квалификатором `const`, который запрещает изменение объекта в этом методе.

5. Метод, позволяющий установить значение элемента

   ```
   void set_item (int, int);
   ```

   Данный метод позволяет установить значение элементу массива. Аргументами функции являются два целочисленных `int` значения: номер элемента и значение, которое необхадимо задать этому элементу.

   Если номер элемента меньше нуля или больше размера массива, то инструкцией `throw` генерируется объект `OutOfRange`.

6. Перегрузка оператора присваивания

   ```
   void operator=(Array &);
   ```

   Оператор присваивания устанавливает объекту значения другого объекта этого класса, переданного по ссылке.

   Перегрузка индексации

```
int operator[](int) const;
```

Аргументом функции является номера элемента, значение которого нам нужно получить. Возращаемым знчением является это значение. Если номер элемента меньше нуля или больше размера массива, то инструкцией `throw` генерируется объект `OutOfRange`. Метод объявлен квалификатором `const`, который запрещает изменение объекта в этом методе.

7. Деструктор

   ```
   ~Array();
   ```

   В деструкторе происходит удалание выделенной памяти под хранение массива с помощью оператора `delete`.

Класс `ArrayApp` было решено спроектировать следующим образом:

1. Поле, содержащее ссылку на объект класса `Array`

   ```
   Array * array;
   ```

2. Конструктор

   ```
   ArrayApp(int size = 10);
   ```

   Пользователь задает размер создаваемого массива. Если же объяект создается без параметров, используется аргумент по умолчанию.

3. Методы, осуществляющие ввод значений

   ```
   void enter_array();
   ```

   В данном случае используется размер массива, который был задан при создании объекта. Пользователю предагается поочередено ввести значения каждого элемета. В случае генерирования исключения во время создания и инциализаци массива, происходит обработка ошибок в блоке `catch`.

   ```
   void enter_array(int);
   ```

   Аргументом данной функции является целочисленное `int` значение, задающее размер массива. В этом случае используется размер массива, переданный аргументом. Пользователю предагается поочередено ввести значения каждого элемета. В случае генерирования исключения во время создания и инциализаци массива, происходит обработка ошибок в блоке `catch`.

4. Метод, выводящий массив в консоль

   `void show_array() const;`

   В этом методе происходит печать размера массива и его содержимого.

5. Деструктор

   `~ArrayApp();`

   В деструкторе происходит удалание выделенной памяти под хранение объекта клас `Array` с помощью оператора `delete`.

5.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.1, компилятор GCC 4.8.4 (x86 64 bit), операционная система Linux Mint 17.2 Cinnamon 64 bit. В процессе выполнения задания производилось ручное тестирование.

5.1.5 Тестовый план и результаты тестирования

Тесты реализованы классом ArrayAppDemo. Листинг приведен в приложении 5.1.7. Все тесты пройдены успешно.

5.1.6 Выводы

При выполнении задания были усвоены навыки проектирования и создания классов и получен опыт в организации многофайлового проекта.

5.1.7 Листинги

array.h

```
#ifndef ARRAY_H
#define ARRAY_H
#include <exception>

class Array
{
private:
    int size;
    int * ptn;
public:
    Array(int size = 10);
    Array(Array &);
    int get_size() const;
    void set_item(const int, const int);
    void operator=(Array &);
    int operator[](int) const;
    ~Array();
};

class OutOfRange: public std::exception
{
public:
    OutOfRange(int index): num(index) {}
    int get_num() const {
        return num;
    }
private:
    int num;
};

#endif // ARRAY_H
```

array.c

```
#include "array.h"
#include "iostream"

Array::Array(int size)
{
    if (size <= 0)
        throw OutOfRange(size);
    else
    {
        this->size = size;
        ptn = new int[this->size];
        for (int i = 0; i < this->size; i++)
```

```
13              ptn[i] = 0;
14      }
15  }
16
17  Array::Array(Array & array)
18  {
19      size = array.get_size();
20      ptn = new int [size];
21      for (int i = 0; i < size; i++)
22          ptn[i] = array[i];
23  }
24
25  int Array::get_size() const
26  {
27      return size;
28  }
29
30  void Array::set_item(const int i, const int item)
31  {
32      if (i < 0 || i >= size)
33          throw OutOfRange(i);
34      else
35          ptn[i] = item;
36  }
37
38  void Array::operator=(Array & array)
39  {
40      delete [] ptn;
41      size = array.get_size();
42      ptn = new int [size];
43      for (int i = 0; i < size; i++)
44          ptn[i] = array[i];
45  }
46
47  int Array::operator[](int i) const
48  {
49      if (i < 0 || i >= size)
50          throw OutOfRange(i);
51      else
52          return ptn[i];
53  }
54
55  Array::~Array()
56  {
57      delete [] ptn;
58  }
```

arrayapp.h

```
1  #ifndef ARRAYAPP_H
```

```
2  #define ARRAYAPP_H
3  #include "array.h"
4  #include "iostream"
5  using namespace std;
6
7  class ArrayApp
8  {
9  private:
10     Array * array;
11 public:
12     ArrayApp(int size = 10);
13     void enter_array();
14     void enter_array(const int);
15     void show_array() const;
16     ~ArrayApp();
17 };
18
19 class InvalidInput: public std::exception
20 {
21 public:
22     InvalidInput(int index): num(index) {}
23     int get_num() const {
24         return num;
25     }
26 private:
27     int num;
28 };
29
30 #endif // ARRAYAPP_H
```

arrayapp.cpp

```
1  #include "arrayapp.h"
2
3  ArrayApp::ArrayApp(int size)
4  {
5      array = new Array(size);
6  }
7
8  void ArrayApp::enter_array()
9  {
10     int item;
11     for (int i = 0; i < array->get_size(); i++)
12     {
13         cout << "#" << i+1 << " ";
14         try {
15             cin >> item;
16             cin.clear();
17             if(cin.get() != '\n')
18                 throw InvalidInput(i+1);
```

```
19          }
20          ///
21          catch (InvalidInput& e)
22          {
23              cout << "#" << e.get_num() << " invalid input, #"
                    << e.get_num() << " = 0" << endl;
24              continue;
25          }
26
27          try {
28              array->set_item(i, item);
29          }
30          catch (OutOfRange& e) {
31              cout << "#" << e.get_num() <<  " out of range" <<
                    endl;
32              continue;
33          }
34      }
35  }
36
37  void ArrayApp::enter_array(int size)
38  {
39      delete array;
40      try {
41          array = new Array(size);
42      }
43      catch (OutOfRange& e) {
44          cout << e.get_num() <<  " invalid size!" << endl;
45          terminate();
46      }
47
48      int item;
49      for (int i = 0; i < size; i++)
50      {
51          cout << "#" << i+1 << " ";
52          try {
53              cin >> item;
54              cin.clear();
55              if(cin.get() != '\n')
56                  throw InvalidInput(i+1);
57          }
58          catch (InvalidInput& e)
59          {
60              cout << "#" << e.get_num() << " invalid input, #"
                    << e.get_num() << " = 0" << endl;
61              continue;
62          }
63
64          try {
```

```
65              array->set_item(i, item);
66          }
67          catch (OutOfRange& e) {
68              cout << "#" << e.get_num() <<  " out of range" <<
                    endl;
69              continue;
70          }
71      }
72  }
73
74  void ArrayApp::show_array() const
75  {
76      int item;
77      cout << "Size = " << array->get_size() << ": [ ";
78      for (int i = 0; i < array->get_size(); i++)
79      {
80          item = array->operator[](i);
81          cout << item << " ";
82      }
83      cout << "]" << endl;
84  }
85
86  ArrayApp::~ArrayApp()
87  {
88      delete array;
89  }
```

Приложения

Листинги модульных тестов к заданиям с 1 по 4 включительно

```
#include <QString>
#include <QtTest>
#include "time.h"
#include "circle_game.h"
#include "inch_to_cm.h"
#include "palindrome.h"
#include "phrases.h"

class TestTest : public QObject
{
    Q_OBJECT
private Q_SLOTS:
    void test_inch_to_cm();
    void test_palindrome();
    void test_time();
    void test_circle_game();
    void test_phrases();
};

void TestTest::test_inch_to_cm()
{
    int inches = 301;
    Meters meter;
    calculating_inch_to_cm(inches, &meter);
    QCOMPARE(meter.m, 7);
    QCOMPARE(meter.cm, 64);
    QCOMPARE(meter.mm, 5.4);
}

void TestTest::test_time()
{
    double velocity[3] = {60, 80, 100};
    double time[3] = {4, 2, 5};
    QCOMPARE(calculating_time(velocity, time), 6.5);
```

```
35 }
36
37 void TestTest::test_palindrome()
38 {
39     char number[20] = "2311441132";
40     QVERIFY2(is_palindrome(number), "Failed palindrome");
41     strcpy(number, "525321");
42     QVERIFY2(!is_palindrome(number), "Failed palindrome");
43 }
44
45 void TestTest::test_circle_game()
46 {
47     QCOMPARE(determine_the_winner(5, 3), 4);
48     QCOMPARE(determine_the_winner(32, 5), 13);
49     QCOMPARE(determine_the_winner(9, 45), 7);
50 }
51
52 void TestTest::test_phrases()
53 {
54     char str[100] = "check.Check check. check. check check.
          Check.check.";
55     upper_case_phrases(str);
56     QCOMPARE(str, "Check.Check check. Check. Check check.
          Check.Check.");
57 }
58
59 QTEST_APPLESS_MAIN(TestTest)
60
61 #include "tst_testtest.moc"
```

Листинги класса Meter

```
#ifndef METER_H
#define METER_H
/* #1.1: INCH TO CM
 * Перевести длину отрезка из дюймов в метры, сантиметры и миллиметры.
 */

class Meter
{
private:
    int inches;
    int m;
    int cm;
    double mm;
    void convert_inch_to_cm();
public:
    Meter();
    int get_inches() const;
    int get_m() const;
    int get_cm() const;
    double get_mm() const;
    void from_inches(const int);
};

#endif // METER_H
```

```
#include "meter.h"


Meter::Meter()
{
    inches = 0;
    m = 0;
    cm = 0;
    mm = 0;
}

int Meter::get_inches() const
{
    return inches;
}

int Meter::get_m() const
{
    return m;
}

```

```
22 int Meter::get_cm() const
23 {
24     return cm;
25 }
26
27 double Meter::get_mm() const
28 {
29     return mm;
30 }
31
32 void Meter::from_inches(int inches)
33 {
34     this->inches = inches;
35     convert_inch_to_cm();
36 }
37
38 void Meter::convert_inch_to_cm()
39 {
40     double total_mm = inches * 25.4;
41
42     m = total_mm / 1000;
43     total_mm -= m * 1000;
44
45     cm = total_mm / 10;
46     total_mm -= cm * 10;
47
48     mm = total_mm;
49 }
```

Листинги класса Route

```
#ifndef ROUTE_H
#define ROUTE_H

/* #1.2: TIME
 * Определить, за какое время путник одолел первую половину пути, двигаясь T1 часов со скоростью V1, T2 часов со скоростью V2, T3 часов со скоростью V3.
 */

#include <vector>
using std::vector;

class Route
{
private:
    static const int NUMBER_OF_PIECES = 3;
    vector<double> velocity;
    vector<double> time;
public:
    Route(const double *, const double *);
    Route(const vector<double> &, const vector<double> &);
    Route(const double, const double, const double, const double, const double, const double);
    double distance() const;
    double time_of_half_way() const;
};

#endif // ROUTE_H
```

```
#include "route.h"

Route::Route(const double velocity[], const double time[])
{
    this->velocity.reserve(NUMBER_OF_PIECES);
    this->time.reserve(NUMBER_OF_PIECES);
    for(int i = 0; i < NUMBER_OF_PIECES; i++)
    {
        this->velocity[i] = velocity[i];
        this->time[i] = time[i];
    }
}

Route::Route(const vector<double> & velocity, const vector<double> & time): velocity(velocity), time(time) {}

Route::Route(const double v1, const double t1, const double v2, const double t2, const double v3, const double t3)
```

```
17 {
18     this->velocity.reserve(NUMBER_OF_PIECES);
19     this->time.reserve(NUMBER_OF_PIECES);
20     velocity[0] = v1;
21     velocity[1] = v2;
22     velocity[2] = v3;
23     time[0] = t1;
24     time[1] = t2;
25     time[2] = t3;
26 }
27
28 double Route::distance() const
29 {
30     double s = 0;
31     int i;
32     for (i = 0; i < NUMBER_OF_PIECES ; i++)
33         s += velocity[i]*time[i];
34     return s;
35 }
36
37 double Route::time_of_half_way() const
38 {
39     double halfdist = distance()/2;
40     double total_time = 0;
41     int i;
42     for (i = 0; i < NUMBER_OF_PIECES; i++)
43     {
44         if (velocity[i]*time[i]<halfdist)
45         {
46             total_time += time[i];
47             halfdist = halfdist - velocity[i]*time[i];
48         }
49         else
50         {
51             total_time += halfdist/velocity[i];
52             halfdist = 0;
53         }
54     }
55     return total_time;
56 }
```

Листинги класса Number

```h
#ifndef NUMBER_H
#define NUMBER_H

/* #2: PALINDROM
 * Проверить, является ли заданное число NUMBER палиндромом (
     симметричным, первая цифра равна последней и так далее).
 */

#include "string"
#include "sstream"
using namespace std;

class Number
{
private:
    string num;
    string int_to_string(const int);
public:
    Number(const int);
    Number(const string);
    bool is_palindrome() const;
    string get_num() const;
};

#endif // NUMBER_H
```

```cpp
#include "number.h"

string Number::int_to_string(int num)
{
    ostringstream temp;
    temp << num;
    return temp.str();
}

Number::Number(int num)
{
    this->num = int_to_string(num);
}

Number::Number(string num)
{
    this->num = num;
}

bool Number::is_palindrome() const
{
```

```
22      int halflen = num.length() / 2;
23      for (int i = 0; i < halflen; i++)
24          if (num[i] != num[num.length()-1-i])
25              return false;
26      return true;
27  }
28
29  string Number::get_num() const
30  {
31      return num;
32  }
```

Листинги класса CircleGame

```
#ifndef CIRCLEGAME_H
#define CIRCLEGAME_H

/* #3: CIRCLE GAME
 * По кругу располагаются NUM_OF_PLAYERS человек. Ведущий счи
    тает по кругу, начиная с первого, и выводит NUM_FOR_KICK-
    го человека.
 * Круг смыкается, счет возобновляется со следующего; так  пр
    одолжается, пока в круге не останется только один человек
    . Найти номер этого человека.
 */

#include <vector>
using std::vector;

class CircleGame
{
private:
    int players;
    int kicking;
public:
    CircleGame(const int, const int);
    void set_players(const int);
    void set_kicking(const int);
    int determine_the_winner() const;
};

#endif // CIRCLEGAME_H
```

```
#include "circlegame.h"

CircleGame::CircleGame(int players, int kicking): players(
   players), kicking(kicking) {}

void CircleGame::set_players(int players)
{
    this->players = players;
}

void CircleGame::set_kicking(int kicking)
{
    this->kicking = kicking;
}

int CircleGame::determine_the_winner() const
{
    vector<int> is_player_in_game(players);
```

```
18      int i;
19      for (i = 0; i < players; i++)
20      {
21          is_player_in_game[i] = 1;
22      }
23      for (int cursor = 0, i = 0; i < players-1; i++) // проигр
            ать должны players-1 игроков
24      {
25          for (int j = 0; j < kicking-1; cursor++) // считаем п
                одряд kicking-1 игроков, остающихся в игре
26              if (is_player_in_game[cursor % players])
27                  j++;
28
29          while (!is_player_in_game[cursor % players]) // ищем
                следующего kicking игрока, который пока в игре
30              cursor++;
31
32          is_player_in_game[cursor % players] = 0; // kicking и
                грок объявляется проигравшим
33          cursor += 1;
34      }
35      for(i = 0; is_player_in_game[i]==0 && i < players; i++);
36      return i + 1;
37  }
```

Листинги класса Phrases

```
#ifndef PHRASES_H
#define PHRASES_H

/* #4: PHRASES
 * Проверить, что все фразы в тексте  начинаются с прописной
     буквы и при необходимости откорректировать текст.
 */

#include "string"
using namespace std;

class Phrases
{
private:
    string str;
public:
    Phrases(const string);
    string get_str() const;
    void upper_case_phrases();
};

#endif // PHRASES_H
```

```
#include "phrases.h"

Phrases::Phrases(string str): str(str) {}

string Phrases::get_str() const
{
    return str;
}

void Phrases::upper_case_phrases()
{
    str[0] = toupper(str[0]);
    for (unsigned i = 0; i < str.length(); i++)
    {
        if (str[i] == '.')
        {
            int beginingOfSentence = i+1;
            if (str[i+1] == ' ')
                beginingOfSentence++;
            str[beginingOfSentence] = toupper(str[
               beginingOfSentence]);
        }
    }
}
```

Листинги модульных тестов к заданиям с 1 по 4 включительно на языке C++

```
#include "tst_cpptesttest.h"

class CpptestTest : public QObject
{
    Q_OBJECT
private Q_SLOTS:
    void array();
    void meter();
    void route();
    void circle_game();
    void number();
    void phrases();
};

void CpptestTest::array()
{
    Array array1(5);
    QCOMPARE(array1.get_size(), 5);
    array1.set_item(1, 3);
    QCOMPARE(array1[1], 3);

    Array array2(array1);
    QCOMPARE(array2.get_size(), 5);
    QCOMPARE(array2[1], 3);

    QVERIFY_EXCEPTION_THROWN( array2.set_item(1000, 1), std::
        exception );
    QVERIFY_EXCEPTION_THROWN( Array array3(-2), std::
        exception );
}

void CpptestTest::meter()
{
    Meter units;
    units.from_inches(301);
    QCOMPARE(units.get_m(), 7);
    QCOMPARE(units.get_cm(), 64);
    QCOMPARE(units.get_mm(), 5.4);
}

void CpptestTest::route()
{
    Route route(50, 1, 100, 1, 150, 1);
    QCOMPARE(route.time_of_half_way(), 2.0);

```

```
44      double velocity[3] = {60, 80, 100};
45      double time[3] = {4, 2, 5};
46      Route route2(velocity, time);
47      QCOMPARE(route2.time_of_half_way(), 6.5);
48
49      vector<double> velo(3);
50      velo.insert(velo.end(), 60);
51      velo.insert(velo.end(), 80);
52      velo.insert(velo.end(), 100);
53      vector<double> tm(3);
54      tm.insert(tm.end(), 4);
55      tm.insert(tm.end(), 2);
56      tm.insert(tm.end(), 5);
57      Route route3(velo, tm);
58      QCOMPARE(route3.time_of_half_way(), 6.5);
59  }
60
61  void CpptestTest::circle_game()
62  {
63      CircleGame game(5, 3);
64      QCOMPARE(game.determine_the_winner(), 4);
65      game.set_players(32);
66      game.set_kicking(5);
67      QCOMPARE(game.determine_the_winner(), 13);
68  }
69
70  void CpptestTest::number()
71  {
72      Number number1("15951");
73      QCOMPARE(number1.is_palindrome(), true);
74      Number number2(15951);
75      QCOMPARE(number1.get_num(), number2.get_num());
76      Number number3(1234513);
77      QCOMPARE(number3.is_palindrome(), false);
78  }
79
80  void CpptestTest::phrases()
81  {
82      string temp = "check.Check check. check. check check.
            Check.check.";
83      Phrases text(temp);
84      text.upper_case_phrases();
85      temp = "Check.Check check. Check. Check check. Check.
            Check.";
86      QCOMPARE(text.get_str(), temp);
87  }
88
89  QTEST_APPLESS_MAIN(CpptestTest)
90
```

```
91 #include "tst_cpptesttest.moc"
```

Листинги класса ArrayAppDemo

```
#ifndef ARRAYAPPDEMO_H
#define ARRAYAPPDEMO_H

#include "arrayapp.h"

using std::cin;
using std::cout;
using std::endl;
using std::exception;

class ArrayAppDemo
{
private:
    void creating_without_parameter() const;
    void creating_with_fix_size() const;
    void creating_with_users_size() const;
    void creating_with_existing_array(ArrayApp &) const;
public:
    void demo() const;
};

#endif
```

```
#include "arrayappdemo.h"

void ArrayAppDemo::demo() const
{
    creating_without_parameter();
    creating_with_fix_size();
    creating_with_users_size();
}
void ArrayAppDemo::creating_without_parameter() const
{
    ArrayApp array;
    cout << "CHECK THE ARRAY" << endl
        << "Creating Array#1 without parameters:" << endl
        << "Array 0" << endl;
    array.show_array();
    cout << endl;

}

void ArrayAppDemo::creating_with_fix_size() const
{
    ArrayApp array;
    cout << "Creating Array#2 with fixed size 3:" << endl
         << "Array#2" << endl;
```

```
25      array.enter_array(3);
26      array.show_array();
27      cout << endl;
28  }
29
30  void ArrayAppDemo::creating_with_users_size() const
31  {
32      ArrayApp array;
33      cout << "Creating Array#3 with user's size:" << endl
34           << "Array#3" << endl
35           << "Enter size of array: ";
36      int num;
37      cin >> num;
38      array.enter_array(num);
39      array.show_array();
40      cout << endl;
41      creating_with_existing_array(array);
42  }
43
44  void ArrayAppDemo::creating_with_existing_array(ArrayApp &
        array2) const
45  {
46      ArrayApp array(array2);
47      cout << "Creating Array#4 using an existing Array#3:" <<
            endl
48           << "Array#4" << endl;
49      array.show_array();
50      cout << endl;
51  }
```