

Программирование

В. В. Дьячков

20 декабря 2015 г.

Оглавление

1	Основные конструкции языка	3
1.1	Задание 1.1. Перевод дюймов в метрическую систему	3
1.1.1	Задание	3
1.1.2	Теоретические сведения	3
1.1.3	Проектирование	3
1.1.4	Описание тестового стенда и методики тестирования	4
1.1.5	Тестовый план и результаты тестирования	5
1.1.6	Выводы	5
1.1.7	Листинги	6
1.2	Задание 1.2. Расчет времени	9
1.2.1	Задание	9
1.2.2	Теоретические сведения	9
1.2.3	Проектирование	9
1.2.4	Описание тестового стенда и методики тестирования	11
1.2.5	Тестовый план и результаты тестирования	11
1.2.6	Выводы	11
1.2.7	Листинги	12
2	Циклы	15
2.1	Задание 2. Палиндром	15
2.1.1	Задание	15
2.1.2	Теоретические сведения	15
2.1.3	Проектирование	15
2.1.4	Описание тестового стенда и методики тестирования	16
2.1.5	Тестовый план и результаты тестирования	16
2.1.6	Выводы	17
2.1.7	Листинги	18
3	Массивы	21
3.1	Задание 3. Игра	21
3.1.1	Задание	21

3.1.2	Теоретические сведения	21
3.1.3	Проектирование	21
3.1.4	Описание тестового стенда и методики тестирования	23
3.1.5	Тестовый план и результаты тестирования	23
3.1.6	Выводы	23
3.1.7	Листинги	24
4	Строки	28
4.1	Задание 4. Фразы	28
4.1.1	Задание	28
4.1.2	Теоретические сведения	28
4.1.3	Проектирование	28
4.1.4	Описание тестового стенда и методики тестирования	29
4.1.5	Тестовый план и результаты тестирования	30
4.1.6	Выводы	30
4.1.7	Листинги	31
5	Инкапсуляция	35
5.1	Задание 5. Массив	35
5.1.1	Задание	35
5.1.2	Теоретические сведения	35
5.1.3	Проектирование	35
5.1.4	Описание тестового стенда и методики тестирования	36
5.1.5	Тестовый план и результаты тестирования	37
5.1.6	Выводы	37
5.1.7	Листинги	38

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 метрический 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, возврат к главному 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 содержатся 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, помогающая 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 5.1 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.

1.1.7 Листинги

inch_to_cm.h

```
1 #ifndef INCH_TO_CM
2 #define INCH_TO_CM
3
4 #ifdef __cplusplus
5 extern "C" {
6 #endif
7
8 typedef struct
9 {
10     int m;
11     int cm;
12     double mm;
13 } Meters;
14
15 void calculating_inch_to_cm(int, Meters *);
16
17 #ifdef __cplusplus
18 }
19 #endif
20
21 #endif // INCH_TO_CM
```

inch_to_cm.c

```
1 #include "inch_to_cm.h"
2
3 void calculating_inch_to_cm(int inches, Meters * meter)
4 {
5     double total_mm = inches * 25.4;
6
7     meter->m = total_mm / 1000;
8     total_mm -= meter->m * 1000;
9
10    meter->cm = total_mm / 10;
11    total_mm -= meter->cm * 10;
12
13    meter->mm = total_mm;
14 }
```

ui_inch_to_cm.h

```
1 #ifndef UI_INCH_TO_CM
2 #define UI_INCH_TO_CM
3
4 #include "inch_to_cm.h"
5
6 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
28                (); break;
29        }
30    }
31    else
32    {
33        puts("Error! Input a number.\n");
34        __fpurge(stdin);
35        menu_inch_to_cm();
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 итерирование по каждому участку пути.

Для решения поставленной задачи были использованы следующие математические факты:

- чтобы найти путь на отдельном участке пути необходимо умножить скорость на данном 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 участка. 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, возврат к главному 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` – вычисленное значение времени.

6. Вспомогательная информация

```
void help_time();
```

Вывод в консоль формулировки задания, помогающая 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 QTest.

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, затраченного 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.

1.2.7 Листинги

time.h

```
1 #ifndef TIME
2 #define TIME
3
4 #ifdef __cplusplus
5 extern "C" {
6 #endif
7
8 #define NUMBER_OF_PIECES 3
9 double halfdistance_time(double *, double *);
10 double calculating_time(double *, double *);
11
12 #ifdef __cplusplus
13 }
14 #endif
15
16 #endif // TIME
```

time.c

```
1 #include "time.h"
2
3 double halfdistance_time(double * velocity, double * time)
4 {
5     double s = 0;
6     int i;
7     for (i = 0; i < NUMBER_OF_PIECES ; i++)
8         s += velocity[i]*time[i];
9     return s/2;
10 }
11
12 double calculating_time(double * velocity, double * time)
13 {
14     double halfdist = halfdistance_time(velocity, time);
15     double total_time = 0;
16     int i;
17     for (i = 0; i < NUMBER_OF_PIECES; i++)
18     {
19         if (velocity[i]*time[i]<halfdist)
20         {
21             total_time += time[i];
22             halfdist = halfdist - velocity[i]*time[i];
23         }
24         else
25         {
26             total_time += halfdist/velocity[i];
27             halfdist = 0;
```

```

28         }
29     }
30     return total_time;
31 }

```

ui_time.h

```

1 #ifndef UI_TIME
2 #define UI_TIME
3
4 #include "time.h"
5
6 void menu_time();
7 void help_time();
8 void input_time();
9 void show_time(double);
10
11 #endif // UI_TIME

```

ui_time.c

```

1 #include "ui.h"
2 #include "time.h"
3 #include "ui_time.h"
4
5 void menu_time()
6 {
7     int num;
8     puts("Calculation time of half way:");
9     puts("1. Input velocity and time");
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_time(); menu_time(); break;
22            case 2:
23                help_time(); menu_time(); break;
24            case 9:
25                main_menu(); break;
26            default:
27                puts("Error! Invalid number.\n"); menu_time();
28                break;
29        }
30    }
31 }

```

```

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, возврат к главному меню или завершение программы.

3. Основное пользовательское взаимодействие

```
void input_palindrome();
```

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

4. Вывод в консоль полученного результата

```
void show_palindrome(char *, int);
```

Параметрами функции являются 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, помогающая 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 5.1 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.

2.1.7 Листинги

palindrome.h

```
1 #ifndef PALINDROME
2 #define PALINDROME
3
4 #ifdef __cplusplus
5 extern "C" {
6 #endif
7
8 #include "string.h"
9
10 int is_palindrome(char *);
11
12 #ifdef __cplusplus
13 }
14 #endif
15
16 #endif // PALINDROME
```

palindrome.c

```
1 #include "palindrome.h"
2
3 int is_palindrome(char * number)
4 {
5     int len = strlen(number);
6     int halflen = len / 2;
7
8     int i = 0;
9     for (i = 0; i < halflen; i++)
10         if (number[i] != number[len - i - 1])
11             return 0;
12     return 1;
13 }
```

ui_palindrome.h

```
1 #ifndef UI_PALINDROME
2 #define UI_PALINDROME
3
4 #include "palindrome.h"
5
6 void menu_palindrome();
7 void input_palindrome();
8 void show_palindrome(char *, int);
9 void help_palindrome();
10
11 #endif // UI_PALINDROME
```

ui_palindrome.c

```
1 #include "ui.h"
2 #include "palindrome.h"
3 #include "ui_palindrome.h"
4
5 void menu_palindrome()
6 {
7     int num;
8     puts("Definition of palindrome:");
9     puts("1. Input a number");
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_palindrome(); menu_palindrome(); break;
22            case 2:
23                help_palindrome(); menu_palindrome(); break;
24            case 9:
25                main_menu(); break;
26            default:
27                puts("Error! Invalid number.\n"); menu_palindrome
28                    (); break;
29        }
30    }
31    else
32    {
33        puts("Error! Input a number.\n");
34        __fpurge(stdin);
35        menu_palindrome();
36    }
37 }
38 void input_palindrome()
39 {
40     char number[20];
41     puts("Input a number:");
42     scanf("%s", number);
43     int is_palin = is_palindrome(number);
44     show_palindrome(number, is_palin);
45     printf("\n");
46 }
```

```

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 взаимодействия пользователя с программой. При помощи цикла `for` и `while` происходит обращение к элементам массива.

Для решения поставленной задачи использовался разработанный алгоритм для определения искомого номера, основанный на итерировании 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, возврат к главному меню или завершение программы.

3. Консольное взаимодействие с пользователем

```
void input_console_circle_game();
```

Пользователю предлагается ввести число, после чего вызывается функция для определения 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 определения 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, помогающая 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 QTest.

3.1.5 Тестовый план и результаты тестирования

В таблице 5.1 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.

3.1.7 Листинги

circle_game.h

```
1 #ifndef CIRCLE_GAME
2 #define CIRCLE_GAME
3
4 #include "stdlib.h"
5
6 #ifdef __cplusplus
7 extern "C" {
8 #endif
9
10 int determine_the_winner(int, int);
11
12 #ifdef __cplusplus
13 }
14 #endif
15
16 #endif // CIRCLE_GAME
```

circle_game.c

```
1 #include "circle_game.h"
2
3 int determine_the_winner(int number_of_players, int
    number_for_kick)
4 {
5     int * is_player_in_game;
6     is_player_in_game = (int *) malloc(sizeof(int) *
    number_of_players);
7     int i;
8     for (i = 0; i < number_of_players; i++)
9     {
10         is_player_in_game[i] = 1;
11     }
12     int j, cursor = 0;
13
14     for (i = 0; i < number_of_players-1; i++) // проиграть до
        лжны number_of_players-1 угров
15     {
16         for (j = 0; j < number_for_kick-1; cursor++) // счита
            ем подряд number_for_kick-1 угров, остающихся в
            угре
17             if (is_player_in_game[cursor % number_of_players
                ])
18                 j++;
19
20         while (is_player_in_game[cursor % number_of_players]
            == 0) // ищем следующего number_for_kick угра, к
```

```

21         оторый пока в игре
           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();
23             break;
24         case 2:
25             input_file_circle_game(); menu_circle_game();
26             break;
27         case 3:
28             help_circle_game(); menu_circle_game(); break;
29         case 9:
30             main_menu(); break;
31         default:
32             puts("Error! Invalid number.\n");
33             menu_circle_game(); break;
34     }
35 }
36 else
37 {
38     puts("Error! Input a number.\n");
39     __fpurge(stdin);
40     menu_circle_game();
41 }
42 }
43
44 void input_console_circle_game()
45 {
46     int n, m;
47     printf("Input the number of players: ");
48     scanf("%d", &n);
49     printf("Input the parameter selection: ");
50     scanf("%d", &m);
51     int winner = determine_the_winner(n, m);
52     show_circle_game(winner);
53     printf("\n");
54 }
55
56 void input_file_circle_game()
57 {
58     int n, m;
59     printf("Write the input file (0 for \"input\"): ");
60     char i_filename[20];
61     scanf("%s", i_filename);
62     if (i_filename[0] == '0')
63         strcpy(i_filename, "input");
64     FILE * read;
65
66     if ((read = fopen(i_filename, "r")) == NULL)
67     {

```

```

65         puts("Error! Cannot open input file. Input
66             interrupted.\n");
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
83                 interrupted.\n");
84             menu_circle_game();
85         }
86         else
87         {
88             int winner = determine_the_winner(n, m);
89             fprintf(write, "#%d won this game", winner);
90             fclose(write);
91             printf("Check result in %s\n\n", o_filename);
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 указатель 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, возврат к главному меню или завершение программы.

3. Консольное взаимодействие с пользователем

```
void input_console_phrases();
```

Пользователю предлагается ввести строку, в которой необходимо проверить правильность написания 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 указатель на строку `char *`, содержащий адрес исправленной строки.

6. Вспомогательная информация

```
void help_phrases();
```

Вывод в консоль формулировки задания, помогающая 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 5.1 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.

4.1.7 Листинги

phrases.h

```
1 #ifndef PHRASES_H
2 #define PHRASES_H
3
4 #ifdef __cplusplus
5 extern "C" {
6 #endif
7
8 #include "stdio.h"
9 #include "ctype.h"
10 void upper_case_phrases(char *);
11
12 #ifdef __cplusplus
13 }
14 #endif
15
16 #endif // PHRASES_H
```

phrases.c

```
1 #include "phrases.h"
2
3 void upper_case_phrases(char * str)
4 {
5     int i;
6     str[0] = toupper(str[0]);
7     for (i = 0; str[i] != '\0'; i++)
8     {
9         if (str[i] == ',')
10         {
11             int beginingOfSentence = i+1;
12             if (str[i+1] == ' ')
13                 beginingOfSentence++;
14             str[beginingOfSentence] = toupper(str[
15                 beginingOfSentence]);
16         }
17     }
```

ui_phrases.h

```
1 #ifndef UI_PHRASES
2 #define UI_PHRASES
3
4 #include "phrases.h"
5 #include <stdio.h>
6 #include "string.h"
7
```



```

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();
31                break;
32        }
33    }
34    else
35    {
36        puts("Error! Input a number.\n");
37        __fpurge(stdin);
38        menu_phrases();
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
65             interrupted.\n");
66         menu_circle_game();
67     }
68     else
69     {
70         fgets(str, 99, read);
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
83                 interrupted.\n");
84             menu_circle_game();
85         }
86         else
87         {
88             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 откорректировать
           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 стандартные функций ввода-вывода языка C++, таких как `cin` и `cout`, объявленные в заголовочном файле *iostream*,

При помощи цикла `for` и происходит итерирование 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 использовался правило русского языка: новое предложение должно начинаться с прописной буквы.

5.1.3 Проектирование

В ходе проектирования было решено выделить 6 функций:

1. Проверка и исправление переданной строки

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

Параметром функции является указатель 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, возврат к главному меню или завершение программы.

3. Консольное взаимодействие с пользователем

```
void input_console_phrases();
```

Пользователю предлагается ввести строку, в которой необходимо проверить правильность написания 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 указатель на строку `char *`, содержащий адрес исправленной строки.

6. Вспомогательная информация

```
void help_phrases();
```

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

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. Модульное тестирование реализовано при помощи фреймворка QtTest.

5.1.5 Тестовый план и результаты тестирования

В таблице 5.1 представлены значения дюймов использованные при тестировании и ожидаемые значения для метров, сантиметров и миллиметров, а также отметка о результате теста.

Таблица 5.1: 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.

5.1.6 Выводы

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

5.1.7 Листинги

array.h

```
1 #ifndef ARRAY_H
2 #define ARRAY_H
3 #include <exception>
4
5 class Array
6 {
7 private:
8     int size;
9     int * ptn;
10 public:
11     Array(int size = 10);
12     Array(Array &);
13     int get_size() const;
14     void set_item(const int, const int);
15     void operator=(Array &);
16     int operator[](int) const;
17     ~Array();
18 };
19
20 class OutOfRange: public std::exception
21 {
22 public:
23     OutOfRange(int index): num(index) {}
24     int get_num() const {
25         return num;
26     }
27 private:
28     int num;
29 };
30
31 class InvalidInput: public std::exception
32 {
33 public:
34     InvalidInput(int index): num(index) {}
35     int get_num() const {
36         return num;
37     }
38 private:
39     int num;
40 };
41
42 #endif // ARRAY_H
```

array.c

```
1 #include "array.h"
```

```

2 #include "iostream"
3
4 Array::Array(int size)
5 {
6     if (size <= 0)
7         throw OutOfRange(size);
8     else
9     {
10         this->size = size;
11         ptn = new int[this->size];
12         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     {

```



```

51         throw OutOfRange(i);
52         return 0;
53     }
54     else
55         return ptn[i];
56 }
57
58 Array::~~Array()
59 {
60     delete [] ptn;
61 }

```

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

```

```

61         }
62
63         try {
64             array->set_item(i, item);
65         }
66         catch (OutOfRangeException& e) {
67             cout << "#" << e.get_num() << " out of range" <<
68                 endl;
69             continue;
70         }
71     }
72
73 void ArrayApp::show_array() const
74 {
75     int item;
76     cout << "Size = " << array->get_size() << ": [ ";
77     for (int i = 0; i < array->get_size(); i++)
78     {
79         item = array->operator[](i);
80         cout << item << " ";
81     }
82     cout << "]" << endl;
83 }
84
85 ArrayApp::~~ArrayApp()
86 {
87     delete array;
88 }

```

Приложения

Листинги модульных тестов к заданиям с 1 по 4 включительно

```
1  #include <QString>
2  #include <QtTest>
3  #include "time.h"
4  #include "circle_game.h"
5  #include "inch_to_cm.h"
6  #include "palindrome.h"
7  #include "phrases.h"
8
9  class TestTest : public QObject
10 {
11     Q_OBJECT
12 private Q_SLOTS:
13     void test_inch_to_cm();
14     void test_palindrome();
15     void test_time();
16     void test_circle_game();
17     void test_phrases();
18 };
19
20 void TestTest::test_inch_to_cm()
21 {
22     int inches = 301;
23     Meters meter;
24     calculating_inch_to_cm(inches, &meter);
25     QCOMPARE(meter.m, 7);
26     QCOMPARE(meter.cm, 64);
27     QCOMPARE(meter.mm, 5.4);
28 }
29
30 void TestTest::test_time()
31 {
32     double velocity[3] = {60, 80, 100};
33     double time[3] = {4, 2, 5};
34     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.
55                     Check.check.";
56     upper_case_phrases(str);
57     QCOMPARE(str, "Check.Check check. Check. Check check.
58                     Check.Check.");
59 }
60
61 QTEST_APPLESS_MAIN(TestTest)
62 #include "tst_testtest.moc"

```