Программирование

В. В. Дьячков

2 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 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 2.1 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 2.1.7.

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

```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 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	f	Модульный	Успешно
50	100	150	1	1	1	f	Ручной	Успешно

Все тесты пройдены успешно. Листинги модульных тестов приведены в приложении 2.1.7.

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

```
#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      printf("Input velocity and time:");
43      int i;
44      for (i = 0; i < NUMBER_OF_PIECES ; i++)
45      {
46          printf("T[%d] = ", i+1);
47          scanf("%lf", &time[i]);
48          printf("\nV[%d] = ", i+1);
49          scanf("%lf", &velocity[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 воспользовался фактом: число называется палиндромом, если первая цифра равна последней, вторая предпоследней и так далее.

2.1.3 Проектирование

В ходе проектирования решено выделить 5 функций:

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

   `oid 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 2.1 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	Является палиндромом	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 2.1.7.

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

```h
#ifndef PALINDROME
#define PALINDROME

#ifdef __cplusplus
extern "C" {
#endif

#include "string.h"

int is_palindrome(char *);

#ifdef __cplusplus
}
#endif

#endif // PALINDROME
```

palindrome.c

```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

```h
#ifndef UI_PALINDROME
#define UI_PALINDROME

#include "palindrome.h"

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

```

```
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
                   (); break;
28         }
29     }
30     else
31     {
32         puts("Error! Input a number.\n");
33         __fpurge(stdin);
34         menu_palindrome();
35     }
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 }
```

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

public:
    TestTest();

private Q_SLOTS:
    void test_inch_to_cm();
    void test_palindrome();
    void test_time();
    void test_circle_game();
    void test_phrases();
};

TestTest::TestTest()
{
}

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);
```

```
35      QCOMPARE(meter.mm, 5.4);
36  }
37
38  /// А есть тест для "не палиндрома"?
39  /// вдруг ваша функция is_palindrome делает всегда return 1?
40  void TestTest::test_palindrome()
41  {
42      char number[20] = "2311441132";
43      QVERIFY2(is_palindrome(number), "Failed palindrome");
44      strcpy(number, "155434551");
45      QVERIFY2(is_palindrome(number), "Failed palindrome");
46  }
47
48  void TestTest::test_time()
49  {
50      double velocity[3] = {50, 100, 150};
51      double time[3] = {1, 1, 1};
52      QCOMPARE(calculating_time(velocity, time), 2.00);
53  }
54
55  void TestTest::test_circle_game()
56  {
57      QCOMPARE(determine_the_winner(5, 3), 4);
58      QCOMPARE(determine_the_winner(3, 5), 1);
59      QCOMPARE(determine_the_winner(32, 5), 13);
60      QCOMPARE(determine_the_winner(9, 45), 7);
61  }
62
63  void TestTest::test_phrases()
64  {
65      char str[100] = "check.Check check. check. check check.
            Check.check.";
66      upper_case_phrases(str);
67      QCOMPARE(str, "Check.Check check. Check. Check check.
            Check.Check.");
68  }
69
70  QTEST_APPLESS_MAIN(TestTest)
71
72  #include "tst_testtest.moc"
```