

Todo list

■ Лучше избегать слова я, лучше писать обезличенные конструкции	1
■ Уместнее, для реализации данной задачи была использована ...	1
■ Поскольку в названиях функций и файлов могут встречаться символы, которые латех считает служебными, лучше использовать команду <code>verb</code> http://tex.stackexchange.com/questions/2790/when-should-one-use-verb-and-when-texttt	1
■ Что значит целостное представление?	1
■ из стандартной библиотеки языка C объявленные в заголовочном файле <code>stdio.h</code>	1
■ скорее не математический, а метрический факт	1
■ Я бы переписала примерно в таком стиле	2
■ Модульные тесты и так автоматические, можно не писать это дополнительно	3
■ Можно сказать полслова про ручное тестирование, и привести скриншот	3
■ Тут уместно сделать таблицу, в которой перечислить все варианты при тестировании	3
■ безличные конструкции	4

Программирование

В. В. Дьячков

28 ноября 2015 г.

Глава 1

Основные конструкции языка

1.1 Задание 1

1.1.1 Задание

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

1.1.2 Теоретические сведения

Лучше избегать слова я, лучше писать обезличенные конструкции

Уместнее, для реализации данной задачи была использована ...

Поскольку в названиях функций и файлов могут встречаться символы, которые латех считает служебными, лучше использовать команду `verb` <http://tex.stackexchange.com/questions/2790/when-should-one-use-verb-and-when-texttt>

Что значит целостное представление?

Для реализации данной задачи я воспользовался структурой `struct` для целостного представления трех величин: метров, сантиметров и миллиметров.

Так же были использованы стандартные функции ввода-вывода `scanf`, `printf`,

из стандартной библиотеки языка C объявленные в заголовочном файле `stdio.h`

`puts` из библиотеки `stdio.h`.

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

скорее не математический, а метрический факт

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

1.1.3 Проектирование

В ходе проектирования решено выделить 5 функций:

1. `calculating_inch_to_cm` — в этой функции реализован перевод дюймов в метры, сантиметры и миллиметры.

В качестве передаваемых параметров используются целочисленное `int` значение дюймов и объявленная в заголовочном файле `inch_to_cm.h` структура `Meters`, передаваемая по ссылке.

Я бы переписала примерно в таком стиле

2. 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`

3. `menu_inch_to_cm` — в этой функции реализованно начальное взаимодействие с пользователем. Пользователю предлагается выбрать консольный ввод, вызов справки, возврат к главному меню или завершение программы.
4. `input_inch_to_cm` — в этой функции реализованно основное взаимодействие с пользователем. Пользователю предлагается ввести дюймы, после чего вызывается функция для перевода в метрическую систему и функция для вывода получившегося результата в консоль.
5. `show_inch_to_cm` — в этой функции происходит вывод в консоль получившегося результата. Для этого в качестве параметров передаются целочисленное `int` значение дюймов, заданное пользователем, и структура `Meters`, в которой находятся результаты вычислений.

6. `help_inch_to_cm` — в этой функции происходит 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.

Модульные тесты и так автоматические, можно не писать это дополнительно

Модульное тестирование реализовано при помощи фреймворка QtTest.

Можно сказать полслова про ручное тестирование, и привести скриншот

1.1.5 Тестовый план и результаты тестирования

При заданном значении `inches = 301` вызывается функция перевода в метрическую систему `calculating_inch_to_cm`, после чего происходит сравнение ожидаемых и реальных результатов при помощи макроса `QCOMPARE`. Тест пройден:

301 дюйм = 7 метров 64 сантиметра 5.4 миллиметра.

Тут уместно сделать таблицу, в которой перечислить все варианты при тестировании

В таблице 1.1 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
10	X	X	X	Ручной	Успешно

Все тесты пройдены успешно. Листинги модульных тестов приведены в приложении 2.2

1.1.6 Выводы

безличные конструкции

При выполнении задания я закрепил свои навыки в работе с основными конструкциями языка С и получил опыт в организации многофайлового проекта и создании модульных тестов.

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 2

1.2.1 Задание

Определить, за какое время путник одолел первую половину пути, двигаясь T_1 часов со скоростью V_1 , T_2 часов со скоростью V_2 , T_3 часов со скоростью V_3 .

1.2.2 Теоритические сведения

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

Так же были использованы стандартные функции ввода-вывода `scanf`, `printf`, `puts` из библиотеки `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 я воспользовался математическими фактами:

- 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 решено выделить 6 функций:

1. `halfdistance_time` — в этой функции реализовано нахождение половины пройденного пути.
В качестве передаваемых параметров используются 2 указателя на тип `double` – **скорости** на участках пути и **время**, затраченное на прохождение каждого участка пути.
2. `calculating_time` — в этой функции реализованно вычисление времени, затраченного на прохождение первой половины пути.

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

3. `menu_time` — в этой функции реализованно начальное взаимодействие с пользователем. Пользователю предлагается выбрать консольный ввод, вызов справки, возврат к главному меню или завершение программы.
4. `input_time` — в этой функции реализованно основное взаимодействие с пользователем. 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. `show_time` — в этой функции происходит вывод в консоль получившегося результата. Для этого в качестве параметров передаются вещественное число `double` – вычисленное значение времени.
6. `help_time` — в этой функции происходит 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.

Автоматическое модульное тестирование реализовано при помощи QTest (Приложение 1).

1.2.5 Тестовый план и результаты тестирования

При заданных значениях `velocity[3] = 50, 100, 150;` и `time[3] = 1, 1, 1;` вызывается функция вычисления времени, затраченного на прохождение первой половины пути `calculating_time`, после чего происходит сравнение ожидаемых и реальных результатов при помощи макроса `QCOMPARE`. Тест пройден:

Искомое время = 2 часа.

1.2.6 Выводы

При выполнении задания я закрепил свои навыки в работе с основными конструкциями языка C и получил опыт в организации многофайлового проекта и создании модульных тестов.

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     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 1

2.1.1 Задание

2.1.2 Теоритические сведения

2.1.3 Проектирование

2.1.4 Описание тестового стенда и методики тестирования

2.1.5 Тестовый план и результаты тестирования

2.1.6 Выводы

Приложения

2.2 Листинги модульных тестов к заданиям с 1 по 4 включительно

```
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    else
32    {
33        puts("Error! Input a number.\n");
34    }
35 }
```

```

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 }

```