Программирование

А. Ю. Ламтев

12 декабря 2015 г.

Глава 1

Основные конструкции языка

1.1 Задание 1

1.1.1 Задание

Пользователь задает сумму денег в рублях, меньшую 100 (например, 16). Определить, как выдать эту сумму монетами по 5, 2 и 1 рубль, израсходовав наименьшее количество монет (например, 3 x 5р + 0 x 2р + 1 x 1р).

1.1.2 Теоретические сведения

При разработке приложения были задействованы следующие конструкции языка: оператор ***switch***, структуры данных ***struct*** – и были использованы функции стандартной библиотеки ***printf()***, ***scanf()*** и ***puts()***, определённые в заголовочном файле ***stdio.h***; ***atoi()***, определённая в ***stdlib.h***.

Я решил, что разменять сумму денег монетами номиналом 5, 2 и 1 руб. можно следующим образом. Необходимо, чтобы монет большего номинала было больше, чем монет меньшего номинала, насколько это возможно. Это послужило основой для реализации алгоритма.

1.1.3 Проектирование

В ходе проектирования было решено выделить пять функций, одна из которых отвечает за логику, а остальные за взаимодействие с пользователем.

1. **Логика**

- ***change_ by_ coins()***

  Эта функция вычисляет результат. Она содержит один целочисленный параметр - сумму денег, которую необходимо разменять. Возвращаемое значение имеет структурный тип, который включает 3 целочисленных поля: число монеток в 5 руб, число монеток в 2 руб и число монеток в 1 руб.

2. Взаимодействие с пользователем

- ***exchange_ output()***

  Эта функция выводит в консоль результат функции ***change_ by_ coins()***. Она содержит один параметр структурного типа, который включает 3 целочисленных поля: число монеток в 5 руб, число монеток в 2 руб и число монеток в 1 руб. Возвращаемое значение имеет тип ***void***.

- ***exchange_ parameters()***

  Эта функция отвечает за взаимодействие с пользователем при чтении данных из параметров командной строки. Она содержит 2 параметра: типа ***int*** - количество аргументов командной строки и типа ***char***** - массив, содержащий эти аргументы. Считывает данные из параметров командной строки. Вызывает функцию ***exchange_ output()***, которая в свою очередь выводит в консоль результат. Возвращает пустое значение.

- ***exchange()***

  Эта функция отвечает за взаимодействие с пользователем в интерактивном режиме. Она не имеет параметров. Выводит в консоль сообщение о том, что нужно ввести число. Осуществляет контролируемый ввод данных. Вызывает функцию ***exchange_ output()***, которая уже и выводит в консоль результат. Возвращает пустое значение.

1.1.4 Описание тестового стенда и методики тестирования

Интегрированная среда разработки: Qt Creator 3.5.0 (opensource)
Компилятор: GCC 4.9.1 20140922 (Red Hat 4.9.1-10)
Операционная система: Debian GNU/Linux 8 (jessie) 32-бита (version 3.14.1)

На всех стадиях разработки приложения проходило тестирование, ручное и автоматическое. Последнее осуществлялось посредством модульных тестов **Qt**, основанных на библиотеке ***QTestLib***.

На финальной стадии был проведён статический анализ с помощью утилиты **cppcheck**

1.1.5 Тестовый план и результаты тестирования

1. **Модульное тестирование Qt**

   I тест :

   Входные данные: 28
   Выходные данные: 5 2 1
   Результат: Тест успешно пройден

   II тест :

   Входные данные: 44
   Выходные данные: 8 0 2
   Результат: Тест успешно пройден

2. **Статический анализ cppcheck**

   Утилита **cppcheck** не выявила ошибок.

1.1.6 Выводы

В ходе выполнения работы я получил опыт создания многомодульного приложения с отделением логики от взаимодействия с пользователем. Укрепил навыки в создании структурных типов. А также научился тестировать программу с помощью модульных тестов и анализировать с помощью утилиты **cppcheck**.

Листинги

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #include "exchange.h"
5 #include "main.h"
6
7 void exchange_output(struct purse coins)
```

```
8  {
9      printf("Пятирублёвых монет: %i\n"
10            "Двухрублёвых монет: %i\n"
11            "Рублёвых монет: %i\n",
12            coins.fives, coins.twos, coins.ones);
13 }
14
15 void exchange(void)
16 {
17     int number;
18     struct purse coins;
19
20     do
21     {
22         puts("Сколько рублей нужно разменять?");
23         scanf("%i", &number);
24     }
25     while (number >= 100);
26
27     coins = change_by_coins(number);
28     exchange_output(coins);
29 }
30
31 void exchange_parameters(int argc, char** argv)
32 {
33     switch (argc)
34     {
35         case 2:
36             exchange();
37             break;
38         case 3:
39         {
40             int num = atoi(argv[2]);
41             struct purse coins = change_by_coins(num);
42             exchange_output(coins);
43             break;
44         }
45         default:
46             put_error;
47             help_exchange();
48             break;
49     }
50 }
```

```
1  #include "exchange.h"
2
3  struct purse change_by_coins(int amount)
4  {
5      struct purse coins;
```

```
6       coins.fives = amount / 5;
7       coins.twos = (amount % 5) / 2;
8       coins.ones = (amount % 5) % 2;
9       return coins;
10  }
```

1.2 Задание 2

1.2.1 Задание

1.2.2 Теоретические сведения

1.2.3 Проектирование

1.2.4 Описание тестового стенда и методики тестирования

1.2.5 Тестовый план и результаты тестирования

1.2.6 Выводы

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