Программирование

А. Ю. Осипов

24 декабря 2015 г.

Оглавление

Глава 1

Основные конструкции языка

1.1 Задание 1

1.1.1 Задание

Пользователь задает номер сегодняшнего дня недели (от 1 до 7, например, 6).

Вывести номер дня недели завтрашнего и вчерашнего дня (например, завтра 7, вчера 5).

1.1.2 Теоретические сведения

Было использованно:

- несколько функций для ввода и вывода информации, прототипы которых находятся в <stdio.h>
- конструкции "if"для решения поставленной задачи

Для решения поставленной задачи оказалось достаточно знание синтаксиса ввода-вывода языка си и элементарных математических операций.

1.1.3 Проектирование

Было решено сделать реализацию программы в одной функции в файле `daynumbers.cpp` :

- `daynumbers` для вычисления номера предыдущего и следующего дня недели (и вывода результата работы в консоль)

1.1.4 Описание тестового стенда и методики тестирования

Использовался Qt Creator 3.5.0 (opensource) с GCC 4.9.1 компилятором Операционная система: Windows 7

Ручное тестирование присутствовало в минимальном объеме. Для статического анализа был использован cppcheckgui версии 1.7.1 . Ошибок и предупреждений не было. Автоматическое тестирование с применением модульных тестов не проводилось.

1.1.5 Выводы

При написании данной программы, возникли проблемы с вынесением функций в отдельные файлы и использованием заголовочных файлов, но они были решены в ходе работы.

1.1.6 Листинги

```
#ifndef DAYNUMBERS
#define DAYNUMBERS

#endif // DAYNUMBERS
int inputday;
int next_number;
int past_number;
#include <stdio.h>
```

```
#include <daynumbers.h>

int daynumbers(int inputday)
{
    if (inputday < 7)
    {
        next_number = inputday + 1;
        past_number = inputday - 1;
        printf("Past day number: %d\nNext day number: %d",
           past_number, next_number);

    }
    if (inputday == 7)
    {
        past_number = inputday - 1;
        next_number = 1;
        printf("Past day number: %d\nNext day number: %d",
           past_number, next_number);
```

```
17      }
18      if (inputday > 7)
19          printf("Incorrect insert");
20      return 0;
21  }
```

1.2 Задание 2

1.2.1 Задание

Задано произвольное натуральное число, например, 5433. Определить, делится ли число нацело на три.

1.2.2 Теоретические сведения

Было использованно:

- конструкция "if"для создания нескольких исходов выполнения программы
- несколько функций для ввода и вывода информации, прототипы которых находятся в <stdio.h>
- функция получения остатка от деления "

Для решения поставленной задачи оказалось достаточно знание синтаксиса ввода-вывода и базовых математических операций языка C.

1.2.3 Проектирование

Было решено сделать реализацию поставленной задачи в одной функции, находящейся в файле `DivideBy3.cpp`:

- `DivideBy3` (функция определяет, делится ли число нацело на 3 и выводит соответствующий результат в консоль)

В данном приложении бизнес-логика не отделялась от взаимодействия с пользователем.

1.2.4 Описание тестового стенда и методики тестирования

Использовался Qt Creator 3.5.0 (opensource) с GCC 4.9.1 компилятором Операционная система: Windows 7

Ручное тестирование проводилось. Для статического анализа был использован cppcheckgui версии 1.7.1 . Ошибок и предупреждений не было. Автоматическое тестирование с использованием модульных тестов отсутствовало.

1.2.5 Выводы

При написании данной программы проблем не возникло из-за ее простоты и малых размеров.

1.2.6 Листинги

```
1 #ifndef DIVIDEBY3
2 #define DIVIDEBY3
3
4 #include <stdio.h>
5
6 #endif // DIVIDEBY3
7
8 int input;
```

```
1  #include <divideby3.h>
2
3  int divideBy3(int insert)
4  {
5      if (insert % 3 == 0)
6          printf("The number is divisible by 3\n");
7      else
8          printf("The number is not divisible by 3\n");
9      return 0;
10 }
```

Глава 2

Циклы

2.1 Задание 1

2.1.1 Задание

Найти N-ю цифру последовательности 11235813213455..., составленной из чисел Фибоначчи. Например, 9-я цифра равна 2. Строковые функции не использовать.

2.1.2 Теоретические сведения

Было использовано:

- цикл for для образования последовательности Фибоначчи
- целочисленный массив для хранения чисел Фибоначчи
- несколько функций для ввода и вывода информации, прототипы которых находятся в <stdio.h>
- функция возведения числа в степень, прототип которой находятся в <math.h>

Для решения поставленной задачи оказалось достаточно знание основ синтаксиса языка C и наличие абстрактного мышления.

Было решено попутно вместе с образованием последовательности Фибоначчи запоминать ее длину, и, когда эта длина станет больше или равна номеру нужного нам элемента, выделять из последовательности целое число и уже внутри него искать нужную цифру функцией `check`

2.1.3 Проектирование

Было решено выделить одну функцию для взаимодействия с пользователем, находящуюся в файле `fibonacci.cpp`:

- `fibonacci` для ввода данных из консоли, построения последовательности Фибоначчи и выделения из нее N-ной цифры

Для бизнес логики было решено выделить одну функцию,находящуюся в файле `check`:

- `check_number_of_number` функция для нахождения длины числа

2.1.4 Описание тестового стенда и методики тестирования

Использовался Qt Creator 3.5.0 (opensource) с GCC 4.9.1 компилятором Операционная система: Windows 7

Ручное тестирование проводилось в небольшом объеме. Для статического анализа был использован cppcheckgui версии 1.7.1 . Ошибок и предупреждений не было. Автоматическое тестирование не проводилось.

2.1.5 Выводы

При написании программы изначально было неясно как выделить именно цифру (а не число) из последовательности. Однако, через некоторое время решение было обнаружено: необходимо работать с длинами чисел вместо того,чтобы работать с самими числами.

2.1.6 Листинги

```
1  #ifndef FIBONACCI
2  #define FIBONACCI
3  int number;
4  int i;
5  int *ArrayOfFibonacci;
6  int zifr;
7  int result;
8  #include <stdio.h>
9  #include <stdlib.h>
10
11 #endif // FIBONACCI
```

```c
#include <fibonacci.h>
#include <check.h>
#include <math.h>


int fibonacci(int N)
{
    int setchik = 2;
    int number = 0;
    ArrayOfFibonacci = (int*) malloc(N * sizeof(int)); //выделяем память для массива
    ArrayOfFibonacci[1] = 1;
    ArrayOfFibonacci[2] = 1;
    i = 3;

    for (i = 3; i <= N; i++)
    {
        ArrayOfFibonacci[i] = ArrayOfFibonacci[i-1] + ArrayOfFibonacci[i-2];
        setchik = setchik + check_nuber_of_number( ArrayOfFibonacci[i]);
        if(setchik >= N)
        {
            number = i;
            i = N+1;
        }
    }

    int x = setchik - N; //позиция цифры с конца


    if (x == 0)
        result = ArrayOfFibonacci[number] % 10;
    else
    if (x == check_nuber_of_number(ArrayOfFibonacci[number]) - 1)
        result = (ArrayOfFibonacci[number] / (pow(10,x)));
    else
        result = ArrayOfFibonacci[number] / (int) pow(10,(x -1)) % 10;

    printf("%d\n", result);
    free(ArrayOfFibonacci);
    system("pause");
    return 0;
}
```

```c
#ifndef CHECK
```

```
2 #define CHECK
3 int check_nuber_of_number(int number);
4
5 #endif // CHECK
```

```
1  int check_nuber_of_number(int number)
2  {
3      int k;
4      k = 1;
5      while (number >= 10)
6      {
7          number = number/10;
8          k = k + 1;
9      }
10     return k;
11 }
```

Глава 3

Массивы

3.1 Задание 1

3.1.1 Задание

В матрице A(m,n) записаны натуральные числа. Найти первую строку, в которой либо все числа – простые, либо все числа – составные. Вывести строку и обнаруженный признак (составные числа/простые числа).

3.1.2 Теоретические сведения

Было использовано:

- функции стандартной библиотеки для динамического выделения памяти из <stdlib.h>
- несколько функций для ввода и вывода информации, прототипы которых находятся в <stdio.h>
- функции работы с памятью, прототипы которых находятся в <stdlib.h>

Потребовалось знания синтаксиса языка C и некоторое представление о двумерных массивах.

Было решено сделать реализацию программы через матрицу (двумерный массив). После образования матрицы в ней в цикле для каждой строки проверялись числа на простоту/сложность и выводился соответствующий результатдля всей строки.

3.1.3 Проектирование

Было решено выделить одну функцию для взаимодействия с пользователем, в файле `matrix.c`:

- `matrix` для взаимодействия с пользователем через консоль и запроса у него размеров матрицы (это основная функция, так как она вызывает функции бизнес-логики)

Для бизнес логики было решено выделить две функции,находящиеся в файлах `matrixmaker.c` и `matrixnumbercheck.c` соответственно:

- `matrixmaker`. В этой функции реализовано построение матрицы и вывод итоговой строки (содержащей только простые/составные числа).

- `matrixnumbercheck` В этой функции осуществляется проверка числа на признак простое/сложное

3.1.4 Описание тестового стенда и методики тестирования

Использовался QtCreator с GCC компилятором Операционная система: Windows 7

Ручное тестирование проводилось в большом размере. Для статического анализа был использован cppcheckgui. Ошибок и предупреждений не было. Автоматическое тестирование с применением модульных тестов не проводилось.

3.1.5 Листинги

```
1  #ifndef MATRIX
2  #define MATRIX
3
4  #include <main.h>
5  #include <matrixmaker.h>
6  #include <stdio.h>
7  int M;
8  int N;
9
10 #endif // MATRIX
```

```c
#include <matrix.h>

//В матрице A(m,n) записаны натуральные числа.
//Найти первую строку, в которой либо все числа     простые,
    либо все числа     составные.
//Вывести строку и обнаруженный признак (составные числа/прос
    тые числа).

void matrix()
{
    puts("Enter a size of matrix (M x N)  (M = rows | N =
        columns) ");
    scanf("%d%d", &M, &N);

    matrixmaker(M,N);
 return;
}
```

```c
#ifndef MARIXMAKER
#define MARIXMAKER

void matrixmaker(int strok, int stolbcov);
int strok;
int stolbcov;
int i;
int j;
int numberoutputstroki;
int result;
int itogostrok;
int q;
int var;
#include <matrixnumbercheck.h>
#include <stdio.h>
#include <stdlib.h>


#endif // MARIXMAKER
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <matrixmaker.h>
#include <matrixcheck.h>


void matrixmaker(int strok, int stolbcov)   //строим матриу
{
    q = 0;
    result = -1;
```

```
11
12      int **A = (int**) malloc(strok *sizeof(int));
13
14      for (i = 0; i < strok; i++)
15          A[i] = (int*) malloc(stolbcov*sizeof(int));          //
                выделение памяти под матрицу
16      puts("Enter matrix values\n");
17      for (i = 0; i < strok; i++)
18      {
19          for (j = 0; j < stolbcov; j++)
20          {
21              scanf("%d", &A[i][j]);
22          }
23      }
24      puts("Your matrix: ");
25
26      for (i = 0; i < strok; i++)
27      {
28          for (j = 0; j < stolbcov; j++)                      // в
                ывод матрицы
29          {
30              printf("%d ", A[i][j]);
31          }
32          printf("\n");
33      }
34      printf("\n");
35
36      puts("Checking process...");
37
38      int result;
39      for (i = 0; i < strok; i++)
40      {
41          q = 0;
42          for (j = 0; j < stolbcov; j++)
43          {
44  //              printf("%d ", A[i][j]);
45              q = q + check(A[i][j]);
46          }
47          if (q == 0)
48          {
49              result = i;
50              var = q;
51              j = stolbcov;
52          }
53          if (q == stolbcov)
54          {
55              result = i;
56              var = q;
57              j = stolbcov;
```

```
58          }
59
60      }
61      puts("\n");
62
63
64      if (var == 0)
65      {
66          puts("There's prime numbers\n");
67
68      for (j = 0; j < stolbcov; j++)
69          printf("%d ", A[result][j]);
70      }
71
72      else if (var == stolbcov)
73      {
74          puts("There's composite numbers\n");
75      for (j = 0; j < stolbcov; j++)
76          printf("%d ", A[result][j]);
77      }
78      else if (result == -1)
79          puts("There's no prime or composite numbers\n");
80      else
81          puts("Something goes wrong\n");
82      puts("\n");
83      system("pause");
84      puts("\n");
85      return;
86  }
```

```
1  #ifndef MATRIXCHECK
2  #define MATRIXCHECK
3
4
5  int check(const int number);
6
7  #endif // MATRIXCHECK
```

```
1  #include <matrixnumbercheck.h>
2  int check(const int number) // функция проверяет простое это
       число или составное
3  {
4      int i;
5      int mas[number + 1];
6      int output = 0;
7      mas[1] = 1;
8      for (i = 1; i < number; i++)
9      {
10         mas[i+1] = mas[i] + 1;
```

```
11 //         printf("%d ", mas[i+1]);
12     }
13     for (i = 2; i < number; i++)
14     {
15         if (number % mas[i] == 0)
16             output = output + 1;
17     }
18     if (output == 0)
19         {
20 //         puts("Prostoe");// 0 возвращается если аргументом э
       той функции является простое число
21         return 0;
22     }
23     else
24     {
25 //         puts("Sostavnoe");
26         return 1;
27         }
28 }
```

Глава 4

Строки

4.1 Задание 1

4.1.1 Задание

В заданном тексте найти самое длинное слово и самое длинное предложение.

4.1.2 Теоретические сведения

Было использовано:

- несколько функций для ввода и вывода информации, прототипы которых находятся в <stdio.h>
- функции работы со строками, прототипы которых находятся в <string.h>

Потребовалось знания синтаксиса языка C и понимание представления строк в языке си.

Программа была реализована с помощью вспомогательных строк, в которую, посредством посимвольного анализа текста, заносилось самое длинное слово и предложение соответственно.

4.1.3 Проектирование

Для решения поставленной задачи было решено выделить две функции, находящиеся в файле `strings.c`:

- `strings` в этой функции реализована логика работы приложения - разбиение текста на слова и предложения.

- `read_word` - эта функция отвечает за посимвольное чтение слов из файла, открытого функцией `strings`

4.1.4 Описание тестового стенда и методики тестирования

Использовался QtCreator с GCC компилятором Операционная система: Windows 7

Ручное тестирование практически отсутствовало. Для статического анализа был использован cppcheckgui. Ошибок и предупреждений не было. Автоматические тестирование с применением моульнх тестов не проводилось.

4.1.5 Выводы

При написании программы возникли проблемы, вызванные незнанием синтаксиса строковых функций из <string.h> и отсутствием опыта работы со строками в целом. Однако, эти проблемы были решены в ходе работы.

4.1.6 Листинги

```
1  #include "PriceList.h"
2
3  PriceList::PriceList(int number_of_goods)
4  {
5        std::string *name = new std::string[number_of_goods];
6        int *price = new int[number_of_goods];
7        for (int i = 1; i <= number_of_goods; i++)
8        {
9            name[i] = " ";
10           price[i] = 0;
11       }
12
13 //     char **name = (char**) new char*[number_of_goods]; //Вы
       деляем память под количество строк
14 //     for (int i = 0; i < number_of_goods; i++)
15 //     {
16 //         name[i]= (char*) new char[20];  //Выделяем память п
       од количество символов в строке для каждой строки в отдель
       ности
17 //     }
18      return;
19
```

```
}
void PriceList::setname(int number, std::string name)
{
    PriceList::name[number] = name;
}

void PriceList::setprice(int number, int price)
{
    PriceList::price[number] = price;
}

std::string PriceList::getname(int number)
{
    return name[number];
}

PriceList::getprice(int number)
{
    return price[number];
}

PriceList::findproductprice(int number)
{
    return price[number];
}

PriceList::productdeleat(int number)
{
    name[number] = " ";
    price[number] = 0;
    return 0;
}

PriceList::~PriceList()
{
    delete []name;
    return;
}
```

```
#ifndef CPP_H
#define CPP_H
#include <string>


class PriceList
{
public:
    void setname(int number, std::string name);
    void setprice(int number, int price);
```

```
11      std::string getname(int number);
12      int getprice(int number);
13      productdeleat(int number);
14      int N;
15  private:
16      int price[100];
17      std::string *name;
18      int number_of_goods;
19      PriceList(int number_of_goods); //конструктор
20      PriceList(); //конструктор без параметров
21      ~PriceList(); //деструктор
22  };
23
24
25  #endif // CPP_H
```

Глава 5

Задание на классы

5.1 Прейскурант

5.1.1 Задание

Реализовать класс ПРЕЙСКУРАНТ (таблица товаров и цен). Требуемые методы: конструктор, деструктор, вставка товара и цены, удаление товара, поиск цены товара.

5.1.2 Теоретические сведения

Было использовано:

- <iostream> и пространство имен std, для взаимодействия с пользователем через консоль.

В ходе работы был реализован класс ПРЕЙСКУРАНТ, со всеми требуемыми методами, которые были реализованы по отдельности.

5.1.3 Проектирование

В классе были реализованы следующие поля:

- `number_of_goods` - переменная, отвечающая за количество товаров в прейскуранте
- `price` - цена товара
- `name` - переменная типа `std::string`, отвечающая за наименование товара

и методы:

- `setprice` - установить цену товара
- `setname` - установить наименование товара
- `getprice` - функция, возвращающая цену товара (поиск цены товара)
- `getname` - функция, возвращающая наименование товара (поиск наименования товара)
- `productdeleat` - функция, удаляющая товар из прайс-листа (в данном случае она очищает значение цены и наименования товара)

5.1.4 Описание тестового стенда и методики тестирования

Использовался QtCreator с GCC компилятором Операционная система: Windows 7

Ручное тестирование присутствовало в небольшом объеме. Для статического анализа был использован cppcheckgui. Все предупреждения были либо исправлены в процессе работы, либо оказались несущественными. Автоматическое тестирование с применением модульных тестов не проводилось.

5.1.5 Выводы

В ходе выполнения возникло много проблем в связи с пониманием класса и синтаксисом языка c++, однако в ходе работы все проблем были решены и исполнитель приобрел ценный опыт работы с классами.

5.1.6 Листинги

```
1 #include "PriceList.h"
2
3 PriceList::PriceList(int number_of_goods)
4 {
5       std::string *name = new std::string[number_of_goods];
6       int *price = new int[number_of_goods];
7       for (int i = 1; i <= number_of_goods; i++)
8       {
9           name[i] = " ";
```

```
10            price[i] = 0;
11        }
12
13 //     char **name = (char**) new char*[number_of_goods]; //Вы
       деляем память под количество строк
14 //     for (int i = 0; i < number_of_goods; i++)
15 //     {
16 //         name[i]= (char*) new char[20];  //Выделяем память п
       од количество символов в строке для каждой строки в отдель
       ности
17 //     }
18      return;
19
20 }
21 void PriceList::setname(int number, std::string name)
22 {
23      PriceList::name[number] = name;
24 }
25
26 void PriceList::setprice(int number, int price)
27 {
28      PriceList::price[number] = price;
29 }
30
31 std::string PriceList::getname(int number)
32 {
33      return name[number];
34 }
35
36 PriceList::getprice(int number)
37 {
38      return price[number];
39 }
40
41 PriceList::findproductprice(int number)
42 {
43      return price[number];
44 }
45
46 PriceList::productdeleat(int number)
47 {
48      name[number] = " ";
49      price[number] = 0;
50      return 0;
51 }
52
53 PriceList::~PriceList()
54 {
55      delete []name;
```

```
56      return;
57  }
```