Программирование

А. В. Власова

25 декабря 2015 г.

Глава 1

Основные конструкции языка

1.1 Задание 1

1.1.1 Задание

Пользователь задает длину отрезка в саженях, аршинах и вершках (например, 8 саженей 2 аршина 11.4 вершка). Определить длину того же отрезка в метрах (в данном случае 19). 1 сажень = 3 аршина = 48 вершков, 1 вершок = 4.445 см.

1.1.2 Теоритические сведения

В процессе написания кода программы были использованы функции ввода/вывода `prinf()`, `f{scanf()` и `{puts()`, содержащиеся в стандартном заголовочном файле `<stdio.h>`.

1.1.3 Проектирование

Среда разработки: QT Creator 3.5.1 (opensource)
Компилятор: GCC
В программе разделены взаимодействие с пользователем и бизнес-логика.`trancfer_to_meter`
считывает из консоли введенное пользователем количество саженей, аршинов и вершков, после чего передает их функции `double trancfer_to_meters().trancfer_`
переводит полученные данные в метры. Окончательный результат выводится в консоль.
Для контроля исправности программы были созданы модульные тесты, расположенные в файле `tst_testtest.cpp`.

1.1.4 Тестовый план и результаты тестирования

При проведении модульного тестирования программа вызывает функцию `void trancfer_test()`. Код тестируется с помощью макроса `QCOMPARE()`, который сравнивает значение, полученное в функции `double trancfer_to_meters()` и ожидаемый результат.

1.1.5 Выводы

Выполнение задания позволило освоить процесс разделения программы на взаимодействие с пользователем и бизнес-логику, а также приобрести некоторые навыки отладки кода и исправления ошибок, выводимых компилятором.

Листинги

```
#include <stdio.h>
#include "trancfer_to_meters.h"
#include "trancfer_to_meter.h"

void trancfer()
{
    double sazhen, arshin, vershok;
    double result1 = 0;
    puts("Введите через пробел количество саженей, аршинов и
       вершков");
    scanf("%lf %lf %lf", &sazhen, &arshin, &vershok);
    result1 = trancfer_to_meters(sazhen, arshin, vershok);
    printf("%lf м", result1);

}
```

```
#ifndef TRANSFER_TO_METERS_H
#define TRANSFER_TO_METERS_H

void trancfer();

#endif // TRANSFER_TO_METERS_H
```

```
#include <stdio.h>
#include "trancfer_to_meter.h"

double trancfer_to_meters (double sazhen, double arshin,
   double vershok)
{
```

```
6       double m;
7       double sm;
8       arshin = sazhen * 3 + arshin;
9       vershok = arshin * 16 + vershok;
10      sm = vershok * 4.445;
11      m = sm/100;
12
13      return(m);
14
15  }
```

```
1   #ifndef TRANCFER_TO_METER_H
2   #define TRANCFER_TO_METER_H
3
4   #ifdef __cplusplus
5   extern "C"{
6   #endif
7   double trancfer_to_meters (double, double, double);
8   #ifdef __cplusplus
9   }
10  #endif
11  #endif // TRANCFER_TO_METER_H
```

1.2 Задание 2

1.2.1 Задание

На шахматной доске стоят черный король и белые ладья и слон (ладья бьет по горизонтали и вертикали, слон — по диагоналям). Проверить, есть ли угроза королю и если есть, то от кого именно. Координаты короля, ладьи и слона вводить целыми числами.

1.2.2 Теоритические сведения

В процессе написания кода программы в файле `search_threat.c` был использован оператор ветвления `if`. Также были задействованы функции ввода/вывода `puts()`, `scanf()` и математическая функция `abs()`.

1.2.3 Проектирование

Среда разработки: QT Creator 3.5.1 (opensource)
Компилятор: GCC
В программе разделены взаимодействие с пользователем и бизнес-логика.

`shahmati_ugroza.c` считывает из консоли введенные пользователем координаты короля, ладьи и слона. Полученные данные передаются функции `int search_threat()`, где происходит поиск угроз королю со стороны слона и ладьи. Найденные угрозы выводятся в консоль.
Для контроля исправности программы были созданы модульные тесты, расположенные в файле `tst_testtest.cpp`.

1.2.4 Тестовый план и результаты тестирования

При проведении модульного тестирования программа вызывает функцию `void search_threat_test()`. Код тестируется с помощью макроса `{QCOMPARE()`, который сравнивает значение, полученное в функции `int search_threat()`, и ожидаемый результат.

1.2.5 Выводы

Выполнение задания позволило освоить процесс разделения программы на взаимодействие с пользователем и бизнес-логику, а также приобрести некоторые навыки отладки кода и исправления ошибок, выводимых компилятором.

Листинги

```
#include <stdio.h>
#include "shahmati_ugroza.h"
#include "search_threat.h"

void search_threat_reading()
{
    int xking, yking, xladya, yladya, xslon, yslon;
    int result;


    puts("Введите через пробел координаты короля, ладьи и слона");
    scanf("%d %d %d %d %d %d", &xking, &yking, &xladya, &yladya, &xslon, &yslon);
    result = search_threat(xking, yking, xladya, yladya, xslon, yslon);
    if (result == 1) printf ("threat from elephant");
    if (result == 2) printf ("threat from rook");
    if (result == 3) printf ("threat from elephant");
}
```

```c
#ifndef SHAHMATI_UGROZA_H
#define SHAHMATI_UGROZA

void search_threat_reading();

#endif // SHAHMATI_UGROZA_H
```

```c
#include <stdio.h>
#include <stdlib.h>
#include "search_threat.h"

int search_threat(int xking, int yking, int xladya, int
    yladya, int xslon, int yslon)
{
     int threat;

    if (xking==xladya) threat = 1;
    if (yking==yladya) threat = 2;
    if ((abs(xking-xslon))==(abs(yking-yslon))) threat = 3;

     return (threat);

}
```

```c
#ifndef SEARCH_THREAT_H
#define SEARCH_THREAT_H

#ifdef __cplusplus
extern "C"{
#endif

int search_threat(int, int, int, int, int, int);
#ifdef __cplusplus
}
#endif
#endif // SEARCH_THREAT_H
```

Глава 2

Циклы

2.1 Задание 3

2.1.1 Задание

Используя ряд $sinx = x - \frac{x3}{3!} + \frac{x5}{5!} - \frac{x7}{7!} + \ldots$, рассчитать значение синуса x с заданной точностью (например, 10^{-3}). Заданную точность считать достигнутой, если очередной элемент ряда меньше заданной точности.

2.1.2 Теоритические сведения

В процессе написания кода программы был использован цикл `do while` для поочередного определения слагаемых. Цикл прекращается при достижении указанной точности. Кроме того, были применены функции ввода/вывода
verb+puts()+, `scanf()` и `printf()`.

2.1.3 Проектирование

Среда разработки: QT Creator 3.5.1 (opensource)
Компилятор: GCC
В программе разделены взаимодействие с пользователем и бизнес-логика. `search_sine.c` считывает из консоли введенные пользователем значение x и точность. Полученные данные передаются функции `double sinx()`, где высчитывается значение `sin(x)`.
Для контроля исправности программы были созданы модульные тесты, расположенные в файле `tst_testtest.cpp`.

2.1.4 Тестовый план и результаты тестирования

При проведении модульного тестирования программа вызывает функцию `void search_sine_test()`. Код тестируется с помощью макроса `QCOMPARE()`, который сравнивает значение, полученное в функции `double sinx()`, и ожидаемый результат.

2.1.5 Выводы

Выполнение задания позволило освоить процесс создания цикла `do while`, а также выяснить пути исправления некоторых ошибок, выводимых комплятором.

Листинги

```
#include <stdio.h>
#include "search_sine.h"
#include "sinx.h"



void search_sine ()
{
    double x, exact = 0;
    double result;
    puts("Введите через пробел значения x и точности");
    scanf("%lf %lf", &x, &exact);
    result = sinx(x, exact);
    printf("sinx = %G", result);

}
```

```
#ifndef SEARCH_SINE_H
#define SEARCH_SINE_H

void search_sine();

#endif // SEARCH_SINE_H
```

```
#include <stdio.h>
#include "sinx.h"

double sinx (double x, double exact)
{
    double summand, sin, i;
    summand=x;
```

```
    sin=summand;
    i=0;
    do {
        i++;
        summand*=((-1)*x*x)/(2*i*(2*i+1));
        sin+=summand;
    }
    while (summand>exact);
    return(sin);

}
```

```
#ifndef SINX_H
#define SINX_H

#ifdef __cplusplus
extern "C"{
#endif

double sinx(double, double);


#ifdef __cplusplus
}
#endif
#endif // SINX_H
```

Глава 3

Массивы

3.1 Задание 4

3.1.1 Задание

Квадрат n x n состоит из прозрачных и непрозрачных маленьких квадратов. Имеется ли хотя бы один просвет по каждому из двух измерений? Вывести координаты каждого просвета.

3.1.2 Теоритические сведения

В процессе написания кода программы для работы с файлом и для динамического выделения памяти были использованы указатели. Кроме того, были применены циклы `while` и `for`, оператор ветвления `if`. Также использовались функции для работы с файлом `fopen()` и `fclose()`, функции выделения и освобождения памяти `{malloc()` и `free()`, функция чтения данных из файла `fscanf()`, функция вывода `printf()`.

3.1.3 Проектирование

Среда разработки: QT Creator 3.5.1 (opensource)
Компилятор: GCC
В программе разделены взаимодействие с пользователем и бизнес-логика. Функция `void file_clear()` считывает из файла данные и записывает их в двумерный массив. Полученные данные передаются функции `void clear()`, где происходит поиск пустых строк и столбцов массива.

3.1.4 Тестовый план и результаты тестирования

После написания программы было проведено ручное тестирование.

3.1.5 Выводы

Выполнение задания позволило освоить процесс динамического выделения памяти и научиться работать с массивами.

Листинги

```
#include <stdio.h>
#include <stdlib.h>
#include "file_clear.h"
void file_clear()
{
    FILE *fp;
    fp=fopen("array.txt","a");
    int i,j,n = 0;
    fscanf(fp, "%d \n", &n);
    int *matrix;
    matrix=(int*) malloc(n*n*sizeof(int));
    for (i=0; i<n; i++)
        for (j=0; j<n; j++)
        {
            fscanf(fp, "%d", (matrix+i*n+j));
        }

    fclose (fp);
    void clear(int n, int* matrix);

}
```

```
#ifndef FILE_CLEAR_H
#define FILE_CLEAR_H

void file_clear();

#endif // FILE_CLEAR_H
```

```
#include <stdio.h>
#include <stdlib.h>
#include "search_clear.h"

void clear(int n, int* matrix)
{
    int i,j;
```

```
    int sum = *matrix;

    for (i=0; i<n; i++)
    {
        sum = *(matrix+i*n);
        j = 1;
        while (j<n)
        sum = sum + *(matrix+i*n+j);
        if (sum == 0) printf("Просвет в %d строке", i);
        sum = 0;
    }

    for (j=0; j<n; j++)
    {
        sum = *(matrix+j);
        i = 1;
        while (i<n)
        sum = sum + *(matrix+i*n+j);
        if (sum == 0) printf("Просвет в %d столбце", j);
        sum = 0;
    }
    free (matrix);
}
```

```
#ifndef SEARCH_CLEAR_H
#define SEARCH_CLEAR_H


#ifdef __cplusplus
extern "C"{
#endif

void clear(int, int*);
#ifdef __cplusplus
}
#endif
#endif // SEARCH_CLEAR_H
```

Глава 4

Строки

4.1 Задание 5

4.1.1 Задание

Имеется большой словарь русских слов. Найти в нем слова-палиндромы, одинаково читающиеся как слева направо, так и справа налево, например, АННА, ШАЛАШ.

4.1.2 Теоритические сведения

В процессе написания кода программы была использована функция поиска длины строки `strlenght()`, функции для работы с файлами `fopen()` и `fclose()`. Также были применены циклы `while` и `for`, оператор ветвления
verb+if+.

4.1.3 Проектирование

Среда разработки: QT Creator 3.5.1 (opensource)
Компилятор: GCC
В программе разделены взаимодействие с пользователем и бизнес-логика. `palindromes_read.c` считывает из файла слова и передает их функции `char search_palindromes()`, где определяется, является слово палиндромом или нет.

4.1.4 Тестовый план и результаты тестирования

После написания программы было проведено ручное тестирование.

4.1.5 Выводы

Выполнение задания позволило освоить некоторые функции заголовочного файла <string.h>.

Листинги

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "palindromes_read.h"
#include "palindromes_search.h"

int file_open()
{
    FILE *file;
    file = fopen("palindromes.txt", "r");
    char *string;
    char palindrom[30];
    int result;
    if (file==0)
    {
        printf("Ошибка \n"); return -1;}
    else printf ("Найденные палиндромы:\n");


    while (1)
    { string = fgets(palindrom, sizeof(palindrom), file);
      if (string == NULL) {
          if(feof(file)!=0)
          {printf ("Конец файла \n");
              break;}
          else
          {printf("Ошибка чтения \n");
              break;}
      }


      else result = search_palindromes(string);
      if (result ==0) printf ("%s \n", string);

    }
    fclose(file);
    return 0;

}
```

```c
#ifndef PALINDROMES_READ_H
```

```
#define PALINDROMES_READ_H

int file_open();

#endif // PALINDROMES_READ_H
```

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <palindromes_search.h>

char search_palindromes(char* string)
{
    int i;
    int lenght;
    int count = 0;
    lenght = strlen(string);
    for (i=0; i<=lenght/2; ++i)
    {
        if (string[i] != string[lenght-i])
            ++count;

    }

    return (count);

}
```

```
#ifndef PALINDROMES_SEARCH_H
#define PALINDROMES_SEARCH_H

char search_palindromes(char*);

#endif // PALINDROMES_SEARCH_H
```

Глава 5

Инкапсуляция

5.1 Задание 6

5.1.1 Задание

Реализовать класс ОЧЕРЕДЬ (целых чисел, неограниченного размера). Требуемые методы: конструктор, деструктор, копирование, встать в очередь, выйти из очереди.

5.1.2 Теоритические сведения

В процессе написания кода программы был использован поток вывода `cout` из заголовочного файла `<iostream>`. Для создания исключений были применены операторы `try`, `throw` и `catch`.

5.1.3 Проектирование

Среда разработки: QT Creator 3.5.1 (opensource)
Компилятор: GCC
Для выполнения задания был выделен класс `Queue`, в котором элементы запоминаются в массив в порядке очереди. При удалении первого элемента все остальные сдвигаются на одну ячейку массива вперед. Новые элементы записываются в конец очереди.
В определении класса были выделены следующие методы:

- конструктор `Queue()`, инициализирующий свойства класса;
- конструктор копирования `Queue(const Queue &otherQueue)`, служащий для инициализации объекта класса посредствомдругого;

- метод `int plus_element(int element)`, добавляющий элементы в очередь;
- метод `int minus_element()`, удаляющий элементы из массива;
- метод `int print_queue()`, выводящий полученный массив в консоль.

5.1.4 Тестовый план и результаты тестирования

После написания программы было проведено ручное тестирование.

5.1.5 Выводы

Выполнение задания позволило понять принцип работы с классами, научиться объявлять и использовать свойства и методы класса, освоить процесс добавления ограничений.

Глава 6

Приложение

Листинги

```
#include <stdio.h>
#include "file_clear.h"
#include "search_sine.h"
#include "shahmati_ugroza.h"
#include "trancfer_to_meters.h"
#include "palindromes_read.h"
#include "main.h"

int main()
{
    puts ("1. transfer to meters");
    puts ("2. search chess threat");
    puts ("3. search sine");
    puts ("4. search clear in array");
    puts ("5. search palindromes");

    int choice;
    scanf ("%d", &choice);
    switch (choice) {
    case 1:
        puts ("transfer to meters");

        trancfer();
        break;




    case 2:
        puts ("search chess threat");

```

```
        search_threat_reading();
        break;



    case 3:
        puts ("search sine");

        search_sine();
        break;



    case 4:
        puts ("search clear in array");

        file_clear();
        break;



    case 5:
        puts ("search palindromes");

        file_open();
        break;


    }
    return (0);

}
```

```
#include <QString>
#include <QtTest>
#include "search_clear.h"
#include "search_threat.h"
#include "sinx.h"
#include "trancfer_to_meter.h"


class TestTest : public QObject
{
    Q_OBJECT

```

```
13
14 private Q_SLOTS:
15     void testCase1();
16     void search_sine_test();
17     void search_threat_test();
18     void trancfer_test();
19 };
20
21
22 void TestTestCase1()
23 {
24     QVERIFY2(true, "Failure");
25 }
26 void search_sine_test()
27 {
28 QCOMPARE(sinx(0.5, 0.001), 0.5);
29 }
30
31 void search_threat_test()
32 {
33 QCOMPARE(search_threat(2, 4, 1, 2, 3, 3), "threat from
       elephant");
34 }
35
36 void trancfer_test()
37 {
38 QCOMPARE(trancfer_to_meters(8, 2, 11.4), 19);
39 }
40
41 QTEST_APPLESS_MAIN(TestTest)
42
43 #include "tst_testtest.moc"
```