Программирование

Кузовкина. Е. О

7 февраля 2016 г.

Оглавление

Глава 1

Основные конструкции языка

1.1 Задание 1

1.1.1 Задание 1.1

Пользователь задает моменты начала и конца некоторого промежутка в часах, минутах, секундах, например, 8:20:35 и 13:15:19. Определить длину промежутка в секундах.

1.1.2 Теоретические сведения

Было использовано:

1) функции для ввода и вывода информации (scanf, printf - библиотеки <stdio.h>)

Для решения данной задачи нужно было использовать стандартную библиотеку ввода и вывода языка C и элементарные математические операции. Для нахождения промежутка времени необходимо было перевести время начала и конца в секунды и найти их разность.

1.1.3 Проектирование

Было выделено две функции:

1) `a_gap_of_time.c` - для ввода данных из консоли (взаимодействие с пользователем)

2) `numbers.c` - для вычисления промежутка времени (бизнес логика)

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

Использовалось ручное тестирование, автоматическое тестирование не проводилось. Ошибок и предупреждений не было.

1.1.5 Тестовый план и результаты тестирования

Все тесты были пройдены успешно: полученный результат, совпал с ожидаемым. Результаты тестирования (например, 7 5 3 - 7 часов, 5 минут, 3 секунды):

1. входные данные - 1 1 1 (начало), 2 2 2 (конец)

   ожидаемые значения - 3661

   полученые значения - 3661

2. входные данные - 4 4 4 (начало), 4 4 4 (конец)

   ожидаемые значения - 0

   полученые значения - 0

1.1.6 Выводы

В ходе написания программы не возникло никаких трудностей.

Листинги

```
#ifndef MAIN
#define MAIN

#include "timetable.h"
#include "first_task.h"
#include "matrix.h"


void strings();

#endif // MAIN
```

```
#include "main.h"

int main (void)
{
    int check;
    printf("Choose program\n");
    scanf("%i",&check);

    switch(check)
    {
    case 1:
        first_task();
        break;

    case 2:
        timetable();
        break;

    case 3:
        matrix();
        break;

    case 4:
        strings();
        break;
    }
    system("pause");
    return 0;
}

#ifndef __FIRST_TASK_H
#define __FIRST_TASK_H
#include <stdio.h>

int hour_beginning, minutes_beginning, seconds_beginning;
int hour_end, minutes_end, seconds_end;
int beginning_number, end_number, final_number;

void first_task();
int input;
```

```
void a_gap_of_time();
int numbers(int beginning_hour, int minutes_beginning, int seconds_beginning, in

void is_it_a_triangle();
int a,b,c;

void what_type_of_triangle_is_it();
int d,e,f;
int max;

#endif // FIRST_TASK
```

```
#include "first_task.h"

void first_task()
{
    printf("Insert a number of the program\n");
    scanf("%d", &input);
    switch(input)
    {
    case 1:
        a_gap_of_time();
        break;

    case 2:
        is_it_a_triangle();
        what_type_of_triangle_is_it();
        break;

    default:
        printf( "Badly insert\n" );
    }

}
```

```
#include "first_task.h"

void a_gap_of_time()
{
```

```
    printf("Insert time of the beginning\n");
    scanf("%d%d%d", &hour_beginning, &minutes_beginning, &seconds_beginning);

    printf("Insert time of the end\n");
    scanf("%d%d%d", &hour_end, &minutes_end, &seconds_end);

    printf("%d", numbers(hour_beginning, minutes_beginning, seconds_beginning, h
}

#include "first_task.h"

int numbers(int hour_beginning, int minutes_beginning, int seconds_beginning, in
{
    beginning_number = hour_beginning * 3600 + minutes_beginning * 60 + seconds_
    end_number = hour_end * 3600 + minutes_end * 60 + seconds_end;

    final_number = end_number - beginning_number;
    return final_number;
}
```

1.2 Задание 1

1.2.1 Задание 1.2

Заданы три целых числа: a, b, c. Определить, могут ли они быть длинами сторон треугольника, и если да, является ли является ли данный треугольник остроугольным, прямоугольным или тупоугольным.

1.2.2 Теоретические сведения

Было использовано:

1) функции для ввода и вывода информации (scanf, printf - библиотеки <stdio.h>)

2) конструкция "if"

Для решения данной задачи нужно было использовать стандартную библиотеку ввода и вывода языка C и умение определять вид треугольника. Для этого необходимо было сравнить длину каждой стороны с суммой двух других сторон и сравнить квадрат длины каждой стороны с суммой квадратов двух других сторон.

1.2.3 Проектирование

Было выделено две функции:

- `is_it_a_triangle.c` - для ввода данных из консоли и проверки, существует ли такой треугольник
- `what_type_of_triangle_is_it.c` - для вывода типа треугольника

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

Использовалось ручное тестирование, автоматическое тестирование не проводилось. Ошибок и предупреждений не было.

1.2.5 Тестовый план и результаты тестирования

Все тесты были пройдены успешно: полученный результат, совпал с ожидаемым. Результаты тестирования:

1. входные данные - 3 4 5

   ожидаемые значения - "Это треугольник. Он правильный"

   полученые "This is a right triangle"

2. входные данные - 1 2 3

   ожидаемые значения - "Это не треугольник"

   полученые "This is not a triangle"

1.2.6 Выводы

При написание данной программы никаких трудностей не было, так как были достаточные знания о геометрических свойствах треугольника.

Листинги

```
#include "first_task.h"

void is_it_a_triangle()
{
```

```
    printf("Insert 3 sites of triangle\n");
    scanf("%d%d%d",&a,&b,&c);

    if ((a >= b+c) || (b >= c+a) || (c >= a+b))
    {
        printf("This is not a triangle\n");
    }
    else
    {
        printf("This is a triangle\n");
    }
}

#include "first_task.h"

void what_type_of_triangle_is_it()
{
    printf("Insert 3 sites of triangle\n");
    scanf("%d%d%d", &d,&e,&f);

    if ((d*d > e*e + f*f) || (e*e > d*d + f*f) || (f*f > d*d + e*e))
    {
        printf("This is an obtuse triangle\n");
    }

    if ((d>e)&&(d>f))
    {
        max = d;
        if  ((max*max < e*e + f*f))
        {
            printf("This is an acute-angled triangle");
        }
    }
    if ((e>d)&&(e>f))
    {
        max = e;

        if  ((max*max < d*d + f*f))
        {
            printf("This is an acute-angled triangle");
        }
```

```
    }
    else
    {
        max = f;

        if ((max*max < d*d + e*e))
        {
            printf("This is an acute-angled triangle");
        }
    }

    if ((d*d == e*e + f*f) || (e*e == d*d + f*f) || (f*f == d*d + e*e))
    {
        printf("This is a right triangle\n");
    }
}
```

Глава 2

Циклы

2.1 Задание 2

2.1.1 Задание

Сформировать расписание звонков в школе. Время начала занятий 9:00. Количество уроков, длительность урока, длительность малого и большого перерыва, число уроков до большого перерыва (он один раз в день) задаются пользователем. Например, урок 1 2 3 4 5 6 начало 9:00 9:55 10:50 11:45 13:00 13:55 конец 9:45 10:40 11:35 12:30 13:45 14:40

2.1.2 Теоретические сведения

Было использовано:

1) функции для ввода и вывода информации (scanf, printf - библиотеки <stdio.h>)

2) конструкция "if"

Для решения данной задачи необходимо понимать структуру школьного расписания.

Было создано расписание с учетом перемен.

2.1.3 Проектирование

Было выделено две функции:

- `timetable.c` для ввода данных из консоли (взаимодействие с пользователем)
- `timetable_work.c` для формирования расписания

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

Использовалось ручное тестирование, автоматическое тестирование не проводилось. Ошибок и предупреждений не было.

2.1.5 Тестовый план и результаты тестирования

Все тесты были пройдены успешно: полученный результат, совпал с ожидаемым. Результаты тестирования:

1. входные данные - 5 40 10 15 3

   ожидаемые - 9:00-9:40, 9:50-10:30, 10:40-11:20, 11:35-12:15, 12:25-13:05
   полученые - 9:00-9:40, 9:50-10:30, 10:40-11:20, 11:35-12:15, 12:25-13:05

2. входные данные - 4 50 10 20 2

   ожидаемые - 9:00-9:50, 10:00-10:50, 11:20-12:10, 12:10-13:00 полученые - 9:00-9:50, 10:00-10:50, 11:20-12:10, 12:10-13:00

2.1.6 Выводы

В ходе написания программы не возникло никаких трудностей.

Листинги

```
#ifndef __TIMETABLE_H
#define __TIMETABLE_H

#include <stdio.h>
#include <math.h>

void timetable(void);
int number_of_lessons, length_of_the_lesson, short_break, long_break, number_of_

void timetable_work(int number_of_lessons, int length_of_the_lesson, int short_b
int i;
int j;

#endif // TIMETABLE
```

```
#include "timetable.h"

void timetable(void)
{
    printf("Insert a number of lessons\n");
    scanf("%d", &number_of_lessons);

    printf("Insert length of the lesson\n");
    scanf("%d", &length_of_the_lesson);

    printf("Insert length of the short break\n");
    scanf("%d", &short_break);

    printf("Insert length of the long break\n");
    scanf("%d", &long_break);

    printf("Insert number of lessons before a long break\n");
    scanf("%d", &number_of_lessons_before_a_long_break);

    timetable_work(number_of_lessons, length_of_the_lesson, short_break, long_br
}
```

```
#include "timetable.h"

void timetable_work(int number_of_lessons, int length_of_the_lesson, int short_b
{

    j = 9*60;
    for (i = 1; i <= number_of_lessons; i++)
    {
        printf("Lesson:%i\n",i);

        if (j%60 < 10)
        {
            printf("Begins:%i:0%i\n", j/60, j%60);
        }

        else printf("Begins:%i:%i\n", j/60, j%60);

        j += length_of_the_lesson;
```

```
        if (j%60 < 10)
        {
            printf("Ends:%i:0%i\n", j/60 ,j%60);
        }

        else printf("Ends:%i:%i\n", j/60, j%60);

        if (i == number_of_lessons_before_a_long_break)
        {
            j += long_break;
        }

        else
            j += short_break;
    }
}
```

Глава 3

Массивы

3.1 Задание 3

3.1.1 Задание

Матрицу K(m,n) заполнить следующим образом. Элементам, находящимся на периферии (по периметру матрицы), присвоить значение 1, периметру оставшейся подматрицы - значение 2 и так далее до заполнения всей матрицы.

3.1.2 Теоретические сведения

Было использовано:

1) функции для ввода и вывода информации (scanf, printf - библиотеки <stdio.h>)

2) конструкция "if"

3) функции работы с динамической памятью (malloc - библиотеки <stdlib.h>)

Для решения данной задачи необходимо уметь работать с динамической памятью.

Нужно было выделить память, посторить матрицу, заполненную по определенному принципу.

3.1.3 Проектирование

Было выделено две функции:

1) `matrix.c` для ввода данных из консоли (взаимодействие с пользователем)

2) `matrix_maker.c` для формирования матрицы

3.1.4 Описание тестового стенда и методики тестирования

Использовался QtCreator с GCC компилятором Операционная система: Windows 7

Использовалось ручное тестирование, автоматическое тестирование не проводилось. Ошибок и предупреждений не было.

3.1.5 Тестовый план и результаты тестирования

Все тесты были пройдены успешно: полученный результат, совпал с ожидаемым.

Полученные результаты: входные данные - 3 2
ожидаемые - 1 1 1 1 1 1
полученые - 1 1 1 1 1 1
входные данные - 5 5
ожидаемые - 1 1 1 1 1 1 2 2 2 1 1 2 3 2 1 1 2 2 2 1 1 1 1 1 1
полученые - 1 1 1 1 1 1 2 2 2 1 1 2 3 2 1 1 2 2 2 1 1 1 1 1 1
входные данные - 5 5
ожидаемые - 1 1 1 1 1 1 1 2 2 2 2 1 1 2 3 3 2 1 1 2 3 3 2 1 1 2 2 2 2 1 1 1 1 1 1 1
полученые - 1 1 1 1 1 1 1 2 2 2 2 1 1 2 3 3 2 1 1 2 3 3 2 1 1 2 2 2 2 1 1 1 1 1 1 1

3.1.6 Выводы

В ходе написания программы не возникло никаких трудностей.

Листинги

```
#ifndef __MATRIX_H
#define __MATRIX_H

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

```
void matrix();
int n;
int m;

int i;


void matrix_maker(int **matrica, int cols, int rows);

#endif // MATRIX

#include "matrix.h"

void matrix(void)
{

    printf("Insert n\n");
    scanf("%i",&n);

    printf("Insert m\n");
    scanf("%i",&m);

    int **matrica=(int **)malloc(m*sizeof(int*));

    for(i = 0; i < m; i++)
        matrica[i] = (int*) malloc (n*sizeof(int));
    matrix_maker(matrica, n, m);
}

#include "matrix.h"


void matrix_maker(int **matrica, int cols, int rows)
{
    int i;
    for(i = 0; i < rows; i++)
    {
        int j;
        for(j = 0; j < cols; j++)
        {
            if (i <= j)
```

```
            matrica[i][j] = i + 1;
        else
            matrica[i][j] = j + 1;
    }
}

if (cols >= rows)
{
    int i;
    for(i = 0; i < rows; i++)
    {
        int j;
        for(j = 0; j < cols; j++)
        {

            if (i + j >= rows)
            {
                if (i >= j)
                    matrica[i][j] = rows - i;
                else
                    matrica[i][j] = cols - j;
            }
            printf("%i ", matrica[i][j]);
        }

        printf("\n");
    }
} else {
    int i;
    for(i = 0; i < rows; i++)
    {
        int j;
        for(j = 0; j < cols; j++)
        {
            if (i + j >= cols)
            {
                if (i > j)
                    matrica[i][j] = rows - i;
                else
                    matrica[i][j] = cols - j;
            }
```

```
                printf("%i ", matrica[i][j]);

            }

            printf("\n");
        }
    }

}
```

Глава 4

Строки

4.1 Задание 4

4.1.1 Задание

Текст содержит следующие знаки корректуры: `$` - сделать красную строку, `#` - удалить следующее слово, `@` - удалить следующее предложение. Произвести указанную корректировку.

4.1.2 Теоретические сведения

Было использовано:

1) функции для ввода и вывода из файла (fopen, fclose- библиотеки <stdio.h>)

2) конструкция "if"

3) конструкция "while"

Для решения данной задачи необходимо было уметь считывать и записывать информацию в файл, делать корректировку текста. Нужно было считывать текст из файла, проверять его на наличие определенных символов и записывать исправленный текст в файл.

4.1.3 Проектирование

- The text of this article is in English. $ See Russian Names in English (Russian Text) for the text in Russian.

Different ways of rendering Russian names into English existed in the past, and several standards of transliteration of Cyrillic into English exist now. As a result, there may be several English spelling variants for the same Russian name or surname: Yulia, Yuliya, Julia, Julja (Юлия); Dmitry, Dmitriy, Dmitri, Dimitri (Дмитрий); Yevgeny, Yevgeniy, Evgeny, Evgeni, Evgeniy, Eugeny (Евгений); Tsvetaeva, Tsvetayeva, Cvetaeva (Цветаева); Zhukovsky, Zhukovski, Zhukovskiy, Jukovsky (Жуковский).

In some cases, the number $ of existing English $ variants is really intimidating. For example, # Муравьёв, a common Russian last name, is represented by more than fifteen variants of spelling in English: Muravyov, Myravyev, Muraviev, Muraviov, Murav'ev, Muravev, Murav'yev, Murav'ov, Muravjov, Muravjev, Mouravieff, Muravieff, Mouravief, Muravief, Muraviof, Muravioff.

Generally, # transliteration of Russian names into Latin is English-oriented now. @ But many Russian names were transliterated according to the French language in the past, and transliteration on the basis of French was the norm for names and surnames in our travel passports until recently. As a result, French-oriented transliteration variants of Russian names are still common. Also, English spelling of Russian names is influenced by tradition and people's personal preferences.

Modern ways of rendering Russian names into English $ try to preserve, as much as possible, both the pronunciation and the recognizable written look of the original Russian name. This material offers examples of typical English spelling variants for Russian names and linguistic recommendations for rendering Russian names into English. Bear in mind that your name should be written in the same way in all of your travel documents because discrepancies may lead to problems when travelling.

4.1.4 Описание тестового стенда и методики тестирования

Использовался QtCreator с GCC компилятором Операционная система: Windows 7 Использовалось ручное тестирование, автоматическое тестирование не проводилось. Ошибок и предупреждений не было.

4.1.5 Тестовый план и результаты тестирования

Все тесты были пройдены успешно: полученный результат, совпал с ожидаемым, то есть была произведена корректировка текста.

Полученный текст: The text of this article is in English. See Russian Names in English (Russian Text) for the text in Russian.

Different ways of rendering Russian names into English existed in the past, and several standards of transliteration of Cyrillic into English exist now. As a result, there may be several English spelling variants for the same Russian name or surname: Yulia, Yuliya, Julia, Julja (Юлия); Dmitry, Dmitriy, Dmitri, Dimitri (Дмитрий); Yevgeny, Yevgeniy, Evgeny, Evgeni, Evgeniy, Eugeny (Евгений); Tsvetaeva, Tsvetayeva, Cvetaeva (Цветаева); Zhukovsky, Zhukovski, Zhukovskiy, Jukovsky (Жуковский).

In some cases, the number of existing English variants is really intimidating. For example, Муравьёв, a common Russian last name, is represented by more than fifteen variants of spelling in English: Muravyov, Myravyev, Muraviev, Muraviov, Murav'ev, Muravev, Murav'yev, Murav'ov, Muravjov, Muravjev, Mouravieff, Muravieff, Mouravief, Muravief, Muraviof, Muravioff.

Generally, transliteration of Russian names into Latin is English-oriented now. . As a result, French-oriented transliteration variants of Russian names are still common. Also, English spelling of Russian names is influenced by tradition and people's personal preferences.

Modern ways of rendering Russian names into English try to preserve, as much as possible, both the pronunciation and the recognizable written look of the original Russian name. This material offers examples of typical English spelling variants for Russian names and linguistic recommendations for rendering Russian names into English. Bear in mind that your name should be written in the same way in all of your travel documents because discrepancies may lead to problems when travelling.

4.1.6 Выводы

В ходе написания программы не возникло никаких трудностей.

Листинги

```
#include <stdio.h>

void read_word(char *s, int *i, char *w)//чтение слова
{
    int j;
    while (s[*i] <= ' ')
        (*i)++;
    j = 0;
    while (s[*i] != '\0' && s[*i] != ' ')
```

```
    {
        w[j] = s[*i];
        j++;
        (*i)++;
    }
    w[j] = '\0';
}
void strings()
{
    FILE *f,*g;
    char s[2000], w[2000], ws[2000], pred[2000], pred1[2000];
    s[0] = '\0';
    ws[0] = '\0';
    w[0] = '\0';
    int vsp=1;
    int sl = 0;
    int str;
    int j = 0;
    f = fopen("2.txt", "r");
    if (!f)
           {
            puts("cannot open the file");
            return;
}
    while (!feof(f))
    {
        fgets(s, 2000, f);

        int i = 0;
        str = 0;
        ws[0] = '\0';
        if (vsp)
        while (s[i] != '\0')
        {
            if (s[i] == '$')
            {
                w[j] = '\n';
                j++;
                w[j] = ' ';
                j++;
                w[j] = ' ';
```

```
            j++;
            w[j] = '\0';
            i++;
        }
        else
        {
            if (s[i] == '#')
                read_word(s, &i, ws);
            else
            {
                if (s[i] == '@')
                {
                    while (s[i + 1] != '.'&&s[i + 1] != '!'&&s[i + 1] != '?'
                        i++;
                    i++;
                    if (s[i] == '\0')
                        vsp = 0;
                }
                else
                {
                    w[j] = s[i];
                    i++;
                    j++;
                    w[j] = '\0';
                }
            }
        }
    }
    else
    {
        while (s[i + 1] != '.'&&s[i + 1] != '!'&&s[i + 1] != '?'&&s[i + 1] !
            i++;
        i++;
        if (s[i] != '\0')
            vsp = 1;
    }

}
printf("text:\n");
puts(w);
fclose(f);
```

```
    remove("2.txt");
    g = fopen("2.txt", "w");
    fputs(w, g);
    fclose(g);
    getchar();
    getchar();
}
```

Глава 5

Задание на классы

5.1 Многочлен

5.1.1 Задание 1

Многочлен Реализовать класс многочлен (в канонической форме - [math]$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$[/math]. Требуемые методы: конструктор, деструктор, копирование, сложение, вычитание, умножение, расчет значения.

5.1.2 Теоретические сведения

Многочлен стандартного вида — это сумма одночленов, составляющих многочлен, записанных в стандартном виде, среди которых нет подобных.

Степень многочлена — наибольшая степень, которую имеют одночлены, составляющие данный многочлен стандартного вида.

Над многочленами можно проводить арифметические операции (сложение, вычитание, умножение, деление).

Было использовано:

1) цикл "for"

2) условие "if...else"

3) динамическое выделение памяти

4) конструкторы

5) деструктор

6) перегрузки операторов

Для решения данной задачи необходимо уметь работать с классами и перегружать операторы. Нужно было создать класс, его поля и методы.

5.1.3 Проектирование

В классе было реализовано: - конструктор по умолчанию, создаёт многочлен второй степени - конструктор с параметром, принимает значение степени многочлена - деструктор - перегрузка оператора + (для сложения многочленов) - перегрузка оператора - (для вычитания многочленов) - перегрузка оператора * (для умножения многочленов) - перегрузка оператора = (для присвоения значения многочлену) - конструктор копирования - функция вывода многочлена - функция получения степени многочлена - функция вычисления значения многочлена при заданном X - функция присвоения значению коэффициента на позиции coeffNumber значения coeffValue

5.1.4 Описание тестового стенда и методики тестирования

Использовался QtCreator с GCC компилятором Операционная система: Windows 7 Использовалось ручное тестирование, автоматическое тестирование не проводилось. Ошибок и предупреждений не было.

5.1.5 Тестовый план и результаты тестирования

Все тесты были пройдены успешно: полученный результат, совпал с ожидаемым.

2 3 4 5 6 7

```
7x^5 + 6x^4 + 5x^3 + 4x^2 + 3x + 2;
7x^5 + 6x^4 + 5x^3 + 5x^2 + 4x + 3;
-7x^5 - 6x^4 - 5x^3 - 3x^2 - 2x - 1;
7x^7 + 13x^6 + 18x^5 + 15x^4 + 12x^3 + 9x^2 + 5x + 2;
```

```
1 1 1 1 1 1
```

```
1x^5 + 1x^4 + 1x^3 + 1x^2 + 1x + 1;
1x^5 + 1x^4 + 1x^3 + 2x^2 + 2x + 2;
-1x^5 - 1x^4 - 1x^3 + 0x^2 + 0x + 0;
1x^7 + 2x^6 + 3x^5 + 3x^4 + 3x^3 + 3x^2 + 2x + 1;
```

```
1 2 1 2 1 2

2x^5 + 1x^4 + 2x^3 + 1x^2 + 2x + 1;
2x^5 + 1x^4 + 2x^3 + 2x^2 + 3x + 2;
-2x^5 - 1x^4 - 2x^3 + 0x^2 - 1x + 0;
2x^7 + 3x^6 + 5x^5 + 4x^4 + 5x^3 + 4x^2 + 3x + 1;
```

5.1.6 Выводы

В ходе написания программы не возникло никаких трудностей

Листинги

```
#ifndef POLYNOMIAL_H
#define POLYNOMIAL_H

class Polynomial
{
public:
    Polynomial();
    Polynomial(unsigned int _polynomialDegree);
    ~Polynomial();

    Polynomial operator +(Polynomial& anotherPolynomial);
    Polynomial operator -(Polynomial& anotherPolynomial);
    Polynomial operator *(Polynomial& anotherPolynomial);
    Polynomial& operator =(const Polynomial& anotherPolynomial);
    Polynomial (const Polynomial& anotherPolynomial);

    void print();
    unsigned int getPolynomialDegree();
    double calculate(double x);
    void setPolynomialCoeff(unsigned int coeffNumber, double coeffValue);

private:
    unsigned int polynomialDegree;
    double *polynomialCoeff;
};

#endif // POLYNOMIAL_H
```

```
#include "polynomial.h"
#include "math.h"
#include <iostream>

Polynomial::Polynomial()
{
    polynomialDegree = 2;
    polynomialCoeff = new double[polynomialDegree + 1];
    for (unsigned int i = 0; i <= polynomialDegree; i++)
        polynomialCoeff[i] = 1;
}

Polynomial::Polynomial(unsigned int _polynomialDegree)
{
    polynomialDegree = _polynomialDegree;
    polynomialCoeff = new double[polynomialDegree + 1];
    for (unsigned int i = 0; i <= polynomialDegree; i++)
        polynomialCoeff[i] = 1;
}

Polynomial::~Polynomial()
{
    delete [] polynomialCoeff;
}

Polynomial Polynomial::operator +(Polynomial& anotherPolynomial)
{
    double *newPolynomialCoeff;
    unsigned int newPolynomialDegree;

    if (polynomialDegree < anotherPolynomial.polynomialDegree)
    {
        newPolynomialCoeff = new double [anotherPolynomial.polynomialDegree + 1]
        newPolynomialDegree = anotherPolynomial.polynomialDegree;

        for (unsigned int i = 0; i <= anotherPolynomial.polynomialDegree; i++)
        {
            if (i <= polynomialDegree)
                newPolynomialCoeff[i] = polynomialCoeff[i] + anotherPolynomial.p
            else
                newPolynomialCoeff[i] = anotherPolynomial.polynomialCoeff[i];
```

```
        }

    } else
    {
        newPolynomialCoeff = new double [polynomialDegree + 1];
        newPolynomialDegree = polynomialDegree;

        for (unsigned int i = 0; i <= polynomialDegree; i++)
        {
            if (i <= anotherPolynomial.polynomialDegree)
                newPolynomialCoeff[i] = polynomialCoeff[i] + anotherPolynomial.p
            else
                newPolynomialCoeff[i] = polynomialCoeff[i];
        }
    }

    Polynomial newPolynomial(newPolynomialDegree);
    for (unsigned int i = 0; i <= newPolynomial.getPolynomialDegree(); i++)
        newPolynomial.setPolynomialCoeff(i, newPolynomialCoeff[i]);

    return newPolynomial;
}

Polynomial Polynomial::operator -(Polynomial& anotherPolynomial)
{
    double *newPolynomialCoeff;
    unsigned int newPolynomialDegree;

    if (polynomialDegree < anotherPolynomial.polynomialDegree)
    {
        newPolynomialCoeff = new double [anotherPolynomial.polynomialDegree + 1]
        newPolynomialDegree = anotherPolynomial.polynomialDegree;

        for (unsigned int i = 0; i <= anotherPolynomial.polynomialDegree; i++)
        {
            if (i <= polynomialDegree)
                newPolynomialCoeff[i] = polynomialCoeff[i] - anotherPolynomial.p
            else
                newPolynomialCoeff[i] = -1 * anotherPolynomial.polynomialCoeff[i
        }
```

```
    } else
    {
        newPolynomialCoeff = new double [polynomialDegree + 1];
        newPolynomialDegree = polynomialDegree;

        for (unsigned int i = 0; i <= polynomialDegree; i++)
        {
            if (i <= anotherPolynomial.polynomialDegree)
                newPolynomialCoeff[i] = polynomialCoeff[i] - anotherPolynomial.p
            else
                newPolynomialCoeff[i] = polynomialCoeff[i];
        }
    }

    Polynomial newPolynomial(newPolynomialDegree);
    for (unsigned int i = 0; i <= newPolynomial.getPolynomialDegree(); i++)
        newPolynomial.setPolynomialCoeff(i, newPolynomialCoeff[i]);

    return newPolynomial;
}

Polynomial Polynomial::operator *(Polynomial& anotherPolynomial)
{
    unsigned int newPolynomialDegree = polynomialDegree + anotherPolynomial.poly
    double *newPolynomialCoeff = new double [newPolynomialDegree + 1];

    for (unsigned int i = 0; i <= newPolynomialDegree; i++)
        newPolynomialCoeff[i] = 0;

    for (unsigned int i = 0; i <= polynomialDegree; i++)
       for (unsigned int j = 0; j <= anotherPolynomial.polynomialDegree; j++)
           newPolynomialCoeff[i + j] += polynomialCoeff[i]*anotherPolynomial.pol

    Polynomial newPolynomial(newPolynomialDegree);

    for (unsigned int i = 0; i <= newPolynomial.getPolynomialDegree(); i++)
        newPolynomial.setPolynomialCoeff(i, newPolynomialCoeff[i]);

    return newPolynomial;
}
```

```
Polynomial& Polynomial::operator =(const Polynomial& anotherPolynomial)
{
    if (&anotherPolynomial == this)
        return *this;

    delete [] polynomialCoeff;
    polynomialDegree = anotherPolynomial.polynomialDegree;
    polynomialCoeff = new double[polynomialDegree + 1];
    for (unsigned int i = 0; i <= polynomialDegree; i++)
        polynomialCoeff[i] = anotherPolynomial.polynomialCoeff[i];

    return *this;
}

Polynomial::Polynomial (const Polynomial& anotherPolynomial)
{
    polynomialDegree = anotherPolynomial.polynomialDegree;

    polynomialCoeff = new double[anotherPolynomial.polynomialDegree + 1];

    for (unsigned int i = 0; i <= anotherPolynomial.polynomialDegree; i++)
        polynomialCoeff[i] = anotherPolynomial.polynomialCoeff[i];
}

void Polynomial::print()
{
    for (int i = polynomialDegree; i >=0; i--)
    {
        std::cout << "(" << polynomialCoeff[i];

        if (i != 0)
            std::cout << "*x^" << i << ") + ";
        else
            std::cout << ")\n";
    }
}

unsigned int Polynomial::getPolynomialDegree()
{
    return this->polynomialDegree;
}
```

```
double Polynomial::calculate(double x)
{
    double polynomialValue = 0;
    for (unsigned int i = 0; i <= polynomialDegree; i++)
    {
        polynomialValue += polynomialCoeff[i]*pow(x, i);
    }

    return polynomialValue;
}

void Polynomial::setPolynomialCoeff(unsigned int coeffNumber, double coeffValue)
{
    if (coeffNumber > polynomialDegree)
        return;

    polynomialCoeff[coeffNumber] = coeffValue;
}

#include <iostream>
#include "polynomial.h"

using namespace std;

int main()
{
    Polynomial pm1;

    cout << "\nPolynomial 1: ";
    pm1.print();

    Polynomial pm2(5);

    cout << "\nPolynomial 2: ";
    pm2.print();

    std::cout << "\nSet polynomial 2 coefficients:\n";
    for (unsigned int i = 0; i <= pm2.getPolynomialDegree(); i++)
    {
        double coeff;
```

```
        std::cin >> coeff;
        pm2.setPolynomialCoeff(i, coeff);
    }

    cout << "\nPolynomial 2 after changing coefficients:\n";
    pm2.print();

    cout << "\nPolynomial 3 = Polynomial 1 + Polynomial 2:\n";
    Polynomial pm3 = pm1 + pm2;
    pm3.print();

    cout << "\nPolynomial 3 = Polynomial 1 - Polynomial 2:\n";
    pm3 = pm1 - pm2;
    pm3.print();

    cout << "\nPolynomial 3 = Polynomial 1 * Polynomial 2:\n";
    pm3 = pm1 * pm2;
    pm3.print();

    cout << "\nPolynomial 3 value in x = 3: ";
    cout << pm3.calculate(3) << endl;

    system("pause");
    return 0;
}
```