**Ф****ЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информационная безопасность»

ОТЧЕТ ПО ПРАКТИКУМУ

по дисциплине

**ПРОГРАММИРОВАНИЕ В СИСТЕМАХ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ**

**Программирование на языке C**

Выполнил студент группы БИН1706:

Зеленохат Р.А.

Москва 2019

# Практикум №1 Компиляция файлов исходного кода и компоновка полученных объектных файлов в исполняемый модуль. Линейные алгоритмы

### Цель практикума

Овладеть навыками создания однофайловых и многофайловых проектов в интегрированной среде разработки и научится создавать линейные программы на языке C с применением арифметических операций.

### Задание

По номеру Вашего варианта выбрать задачу, решаемую в этой лабораторной работе, и выполнить для нее следующие задания.

### Индивидуальный вариант задания

### Решение заданий

#### Задание 1

Составить программу, имеющую линейный алгоритм и состоящую из одной функции void main(). Программу записать в файл с именем task1.c. Скомпилировать, скомпоновать и выполнить.

В функции main организовать вычисление задачи вашего варианта дважды:

* для исходных данных, значения которых задать в виде констант в тексте функции main;
* для исходных произвольных данных, значения которых пользователь Вашей программы должен ввести с клавиатуры в процессе выполнения программы.

Проанализировать результаты работы и сделать выводы. Перенести полученные результаты в отчет.

Вывод действительных чисел осуществлять с точностью до 0.0001

# include <stdio.h>

# include <math.h>

# define \_USE\_MATH\_DEFINES

void main ()

{

float x, f;

x = 5;

f = 2 \* pow ( sin ( 3 \* M\_PI - 2 \* x) , 2) \* pow ( cos ( 5 \* M\_PI + 2 \* x) , 2);

printf ( "%.0f\n", x);

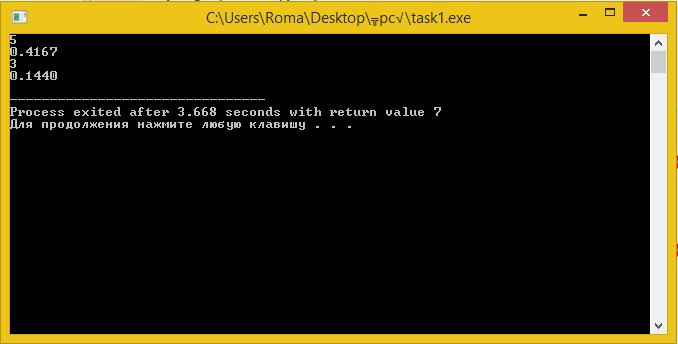
printf ( "%.4f\n", f);

scanf ( "%f" , & x);

f = 2 \* pow ( sin ( 3 \* M\_PI - 2 \* x) , 2) \* pow ( cos ( 5 \* M\_PI + 2 \* x) , 2);

printf ( "%.4f\n", f);

}



#### Задание 2

Линейный алгоритм функции из задания 1 разделить на две процедуры, выделив в одну вычислительные операции этого алгоритма, а в другую все операции ввода-вывода. Каждую процедуру оформить как функцию. Вычислительную часть алгоритма оформить как функцию с параметрами, передаваемыми по значению, и возвращаемым значением. Прототип функции:

double f(double x) // Если функция имеет один параметр

double f(double x, double y) // Если функция имеет два параметра

Другую часть алгоритма оформить как функцию void main(), вызывающую первую функцию нужное количество раз. Записать тексты функций файл с именем task2.с в следующем порядке: функция с параметрами, функция main. Cкомпилировать, скомпоновать и выполнить.

# include <stdio.h>

# include <math.h>

# define \_USE\_MATH\_DEFINES

float f ( float x)

{

float y;

y = 2 \* pow ( sin ( 3 \* M\_PI - 2 \* x) , 2) \* pow ( cos ( 5 \* M\_PI + 2 \* x) , 2);

return y;

}

void main ()

{

float x , res;

scanf ( "%f" , & x);

res = f(x);

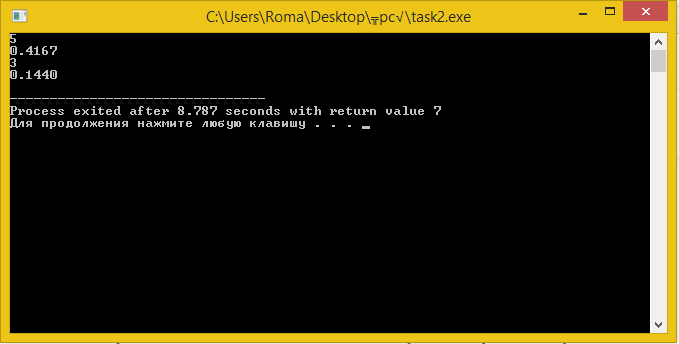
printf ( "%.4f\n", res);

scanf ( "%f" , & x);

res = f(x);

printf ( "%.4f\n", res);

}



Результаты 1 и 2 Задания совпадают.

#### Задание 3

Создать файл task3.с, в котором изменить порядок записи текстов функций, созданных в задании 2. Функции записать в следующем порядке: функция main, функция с параметрами, организующая вычисления (double f(double x) или double f(double x, double y)). Внести требуемые дополнения, добиться успешной компиляции, скомпоновать и выполнить.

# include <stdio.h>

# include <math.h>

# define \_USE\_MATH\_DEFINES

float f ( float x);

void main ()

{

float x , res;

scanf ( "%f" , & x);

res = f(x);

printf ( "%.4f\n", res);

scanf ( "%f" , & x);

res = f(x);

printf ( "%.4f\n", res);

}

float f ( float x)

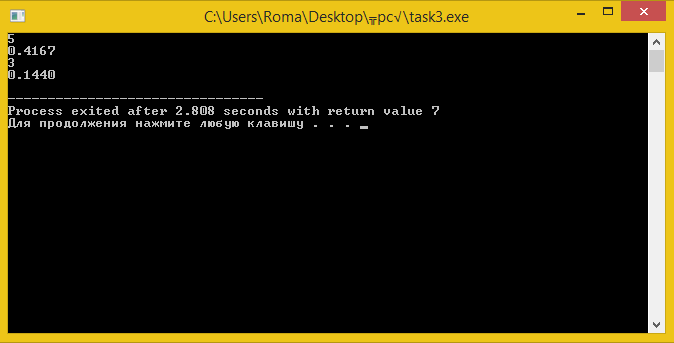
{

float y;

y = 2 \* pow ( sin ( 3 \* M\_PI - 2 \* x) , 2) \* pow ( cos ( 5 \* M\_PI + 2 \* x) , 2);

return y;

}



Результаты аналогичны

#### Задание 4

По-прежнему линейный алгоритм функции из задания 1 разделить на две процедуры, выделив в одну вычислительные операции этого алгоритма, а в другую все операции ввода-вывода.

Вычислительную часть алгоритма оформить как функцию без параметров и без возвращаемого значения. Прототип функции void f(void)

Другую часть алгоритма оформить как функцию void main(), вызывающую вычислительную функцию. Обмен данными между функциями организовать через глобальные объекты (double x, y – аргументы, double result – результат вычисления). Записать тексты функций в файл с именем task4.с в следующем порядке: функция main, затем функция без возвращаемого значения и без параметров. Cкомпилировать, скомпоновать и выполнить.

# include <stdio.h>

# include <math.h>

# define \_USE\_MATH\_DEFINES

float x , res;

void main ()

{

scanf ("%f" , &x);

f();

printf ( "%.4f\n" , res);

scanf ("%f" , &x);

f();

printf ( "%.4f\n", res);

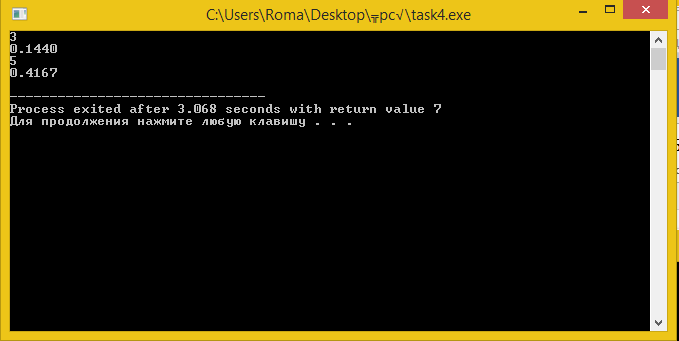
}

void f (void)

{

res = 2 \* pow ( sin ( 3 \* M\_PI - 2 \* x) , 2) \* pow ( cos ( 5 \* M\_PI + 2 \* x) , 2);

}



Результаты аналогичны

#### Задание 5

В этом задании необходимо разделить текст файла task3.с на два файла. В первый файл с именем task5\_main.c поместить текст функции main. Скомпилировать только файл task5\_main.c.

Во второй файл с именем task5\_func.с поместить текст функции с параметрами (double f(double x) или double f(double x, double y)). Скомпилировать только файл task5\_func.c. После раздельной компиляции осуществить совместную компоновку. Полученный исполняемый файл выполнить.

Проанализировать результаты работы и сделать выводы.

Task5\_main

# include <stdio.h>

# include <math.h>

# define \_USE\_MATH\_DEFINES

# include "Task5\_func.c"

float f ( float x);

void main ()

{

float x , res;

scanf ( "%f" , & x);

res = f(x);

printf ( "%.4f\n", res);

scanf ( "%f" , & x);

res = f(x);

printf ( "%.4f\n", res);

}

Task5\_func

# include <stdio.h>

# include <math.h>

# define \_USE\_MATH\_DEFINES

float f ( float x)

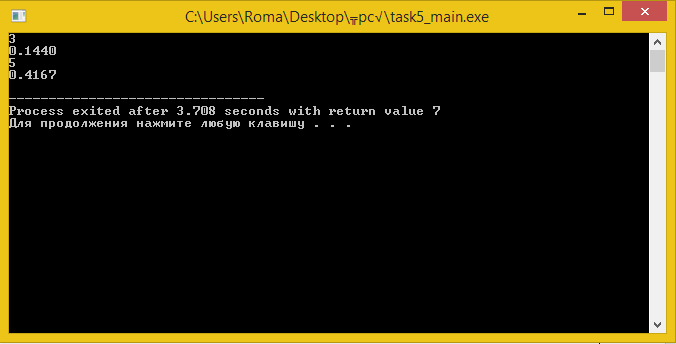
{

float y;

y = 2 \* pow ( sin ( 3 \* M\_PI - 2 \* x) , 2) \* pow ( cos ( 5 \* M\_PI + 2 \* x) , 2);

return y;

}



Результаты аналогичны

#### Задание 6

В этом задании необходимо разделить текст файла task4.с на два файла.

В первый файл с именем task6\_main.c поместить текст функции main. Скомпилировать только файл task6\_main.c.

Во второй файл с именем task6\_func.с поместить определения глобальных объектов (double x, y – аргументы, double result – результат вычисления) и текст функции без параметров (void f(void)). Скомпилировать только файл task6\_func.c.

После раздельной компиляции осуществить совместную компоновку. Разобраться в проблемах, возникающих при совместной компиляции и компоновки. Полученный исполняемый файл выполнить.

Проанализировать результаты работы и сделать выводы.

Task6\_main

# include <stdio.h>

# include <math.h>

# define \_USE\_MATH\_DEFINES

# include "Task6\_func.c"

void main ()

{

scanf ("%f" , &x);

f();

printf ( "%.4f\n" , res);

scanf ("%f" , &x);

f();

printf ( "%.4f\n", res);

}

Task6\_func

# include <stdio.h>

# include <math.h>

# define \_USE\_MATH\_DEFINES

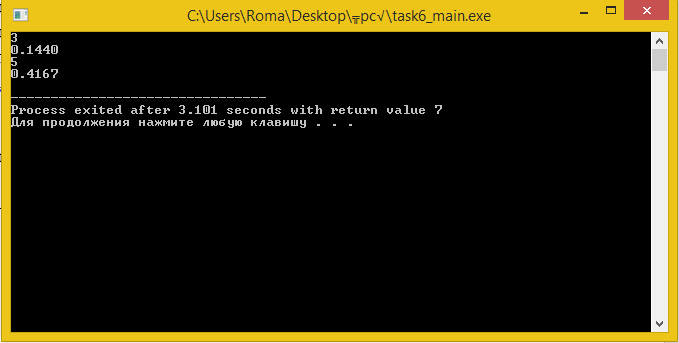
float x , res;

void f (void)

{

res = 2 \* pow ( sin ( 3 \* M\_PI - 2 \* x) , 2) \* pow ( cos ( 5 \* M\_PI + 2 \* x) , 2);

}



Результат аналогичен

#### Задание 7

В этом задании необходимо модифицировать тексты файлов из задания 6.

Описание функции и внешних переменных выделить в отдельный заголовочный файл func.h, включить его содержимое в файлы task7\_main.c и task7\_func.c. Определение внешних переменных произвести в файле task7\_func.c.

Скомпилировать по отдельности файлы task7\_main.c и task7\_func.c. Произвести компоновку, выполнить полученный исполняемый файл.

Task7\_main

# include <stdio.h>

# include <math.h>

# define \_USE\_MATH\_DEFINES

# include "func.h"

# include "Task7\_func.c"

void main ()

{

scanf ("%f" , &x);

f();

printf ( "%.4f\n" , res);

scanf ("%f" , &x);

f();

printf ( "%.4f\n", res);

}

Task7\_func

# include <stdio.h>

# include <math.h>

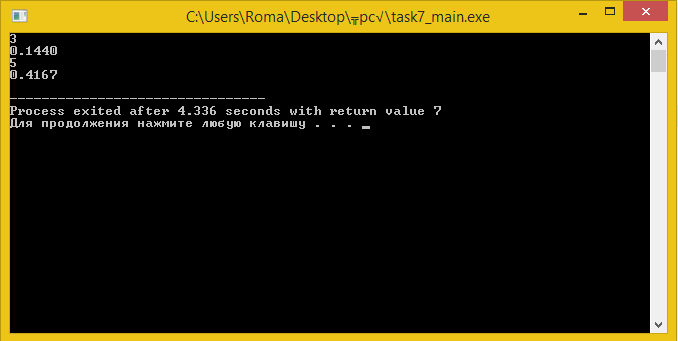
# define \_USE\_MATH\_DEFINES

void f (void)

{

res = 2 \* pow ( sin ( 3 \* M\_PI - 2 \* x) , 2) \* pow ( cos ( 5 \* M\_PI + 2 \* x) , 2);

}



Результаты аналогичны

#### Задание 8

Скомпилировать по отдельности файлы из задания 7 task7\_main.c и task7\_func.c. Создать статическую библиотеку task8\_lib.lib, включающую объектный файл task7\_func.obj. Полученную статическую библиотеку скомпоновать с файлом task7\_main.obj. Выполнить полученный исполняемый файл.

|  |
| --- |
| cmake\_minimum\_required (VERSION 3.8) |
|  |  |
|  | add\_executable(Task8 Task7\_main.c) |
|  |  |
|  | add\_library(Task8\_lib STATIC Task7\_func.c) |
|  |  |
|  | target\_link\_libraries (Task8 Task8\_lib) |

## **Практикум №2** Разветвляющиеся алгоритмы

### Цель практикума

Овладеть навыками создания разветвляющихся алгоритмов на языке C с применением инструкций выбора, условных выражений, логических операций и операций отношения

### Задание

По номеру Вашего варианта (1) выбрать задачи и выполнить следующие задания.

#### Задание 1

Написать функцию, которая получает в качестве параметров координаты точки и определяет, попадает ли она в заштрихованную область на рисунке, который соответствует Вашему варианту (см. таблицу 1). Попадание на границу области считать попаданием в область.

Функция возвращает 1, если точка попадает в область, 0 в противном случае.

Прототип функции \_Bool isInArea(double x, double y).

Функция не должна использовать функции консольного ввода-вывода.

# include <stdio.h>

float Pr1 (float x , float y)

{

int v;

if ( x <= 0 && x >= -1 && y <= 1 && y >= -1 && y >= x || x >= 0 && x <= 1 && y <= 1 && y >= 0 && y >= x )

{

v ++ ;

}

return v;

}

#### Задание 2

Используя условную операцию написать функцию для вычисления выражения, указанного в таблице 2.

Функция получает в качестве входного параметра и возвращает значение выражения.

Прототип функции double f(double x).

Функция не должна использовать функции консольного ввода-вывода.

# include <stdio.h>

# include <math.h>

float f (float x)

{

float y;

if (x <= 3)

{

y = pow (x,2) - 3 \* x + 9;

}

else

{

y = 1 / ( pow(x,3)+6);

}

return y;

}

#### Задание 3

Написать функцию main(), которая будет выводить меню и ожидать ответа пользователя:

1. Задание 1
2. Задание 2

Обработку ответа пользователя осуществить с помощью инструкции switch. При вводе пользователем цифры 1 перейти к демонстрации задания 1. При вводе пользователем цифры 2 перейти к демонстрации задания 2.

Для демонстрации задания 1 запросить у пользователя ввод двух чисел, вызвать разработанную в задании 1 функцию и вывести на экран результат – попадает ли точка в заданную область или нет.

Для демонстрации задания 2 запросить у пользователя ввод числа , вызвать разработанную в задании 2 функцию и вывести результат расчёта на экран.

Скомпилировать все файлы, скомпоновать и выполнить полученный исполняемый файл.

# include <stdio.h>

# include <math.h>

# include "Lab\_21.c"

# include "Lab\_22.c"

float f ( float x);

float Pr1 ( float x, float y);

int main()

{

float x , y;

int res;

int r;

scanf("%d", & r);

switch (r)

{

case 1:

scanf("%f", & x);

scanf("%f", & y);

res = Pr1 (x , y);

printf ("%d", res);

break;

case 2:

scanf("%f", &x);

y = f(x);

printf("%f",y);

break;

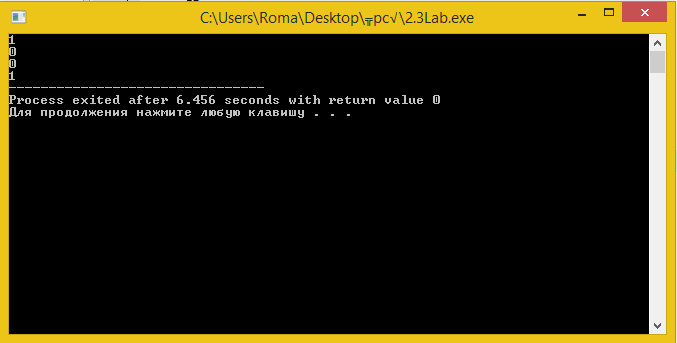
default :

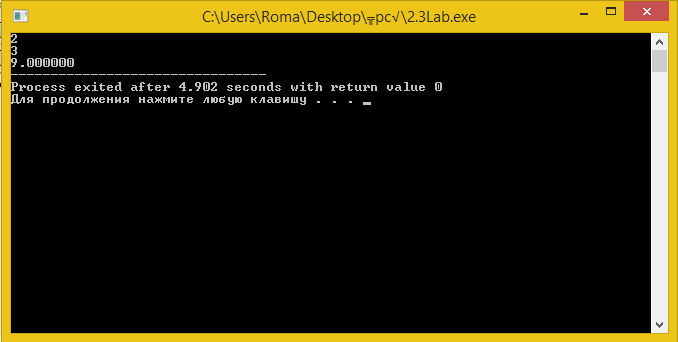
printf ("Error");

}

return 0;

}





Результаты совпадают с действительностью

## Практикум №3 Регулярные и итерационные циклы

### Цель практикума

Овладеть навыками создания циклических алгоритмов на языке C с применением инструкций цикла на примере алгоритмов работы с последовательностями чисел

#### Задание 1

Написать функцию для вычисления выражения .

Функция в качестве параметра принимает значение и вычисляет сумму слагаемых.

Прототип функции double summ(int n).

Функция не должна использовать функции консольного ввода-вывода.

// for

#include <stdio.h>

#include <math.h>

double summ(int n)

{

double a = 0;

double s = 0;

int i;

for (i = 0; i < n - 1; ++i)

{

a = pow(-1,i)\*((double)(pow(i,2)+1)/(double)(pow(i,3)+3));

s += a;

}

return s;

}

// while

#include <stdio.h>

#include <math.h>

double summ(int n)

{

double a = 0;

double s = 0;

int i = 0;

while (i < n - 1)

{

a = pow(-1,i)\*((double)(pow(i,2)+1)/(double)(pow(i,3)+3));

s += a;

++i;

}

return s;

}

#### Задание 2

// do while

#include <stdio.h>

#include <math.h>

double summ(int n)

{

double a = 0;

double s = 0;

int i = 0;

do

{

a = pow(-1,i)\*((double)(pow(i,2)+1)/(double)(pow(i,3)+3));

s += a;

++i;

} while (i < n - 1);

return s;

}

Написать функцию для вычисления выражения с точностью .

Функция в качестве параметра принимает значение и вычисляет сумму до тех пор, пока не выполнится условие .

Прототип функции double summ2(double eps).

Функция не должна использовать функции консольного ввода-вывода.

// for

#include <stdio.h>

#include <math.h>

double summ2(double eps)

{

double a = 0;

double s = 0;

int i;

for (i = 0; ; ++i)

{

a = pow(-1,i)\*((double)(pow(i,2)+1)/(double)(pow(i,3)+3));

if (fabs(a) <= eps)

{

return s;

}

s += a;

}

}

// while

#include <stdio.h>

#include <math.h>

double summ2(double eps)

{

double a = 0;

double s = 0;

int i = 0;

while (1)

{

a = pow(-1,i)\*((double)(pow(i,2)+1)/(double)(pow(i,3)+3));

if (fabs(a) <= eps)

{

return s;

}

s += a;

++i;

}

}

// do while

#include <math.h>

double summ2(double eps)

{

double a = 0;

double s = 0;

int i = 0;

do

{

a = pow(-1,i)\*((double)(pow(i,2)+1)/(double)(pow(i,3)+3));

if (fabs(a) <= eps)

{

return s;

}

s += a;

++i;

} while (1);

}

#### Задание 3

Написать функцию, которая печатает членов последовательности , исключая из неё каждый -ый член.

Числа и передаются в функцию в качестве параметра. Для исключения члена последовательности используйте инструкцию перехода continue.

Прототип функции void print(int n, int k).

Пример вывода: 1.2 1.3 0.75

// for

#include <stdio.h>

#include <math.h>

void print(int n, int k)

{

double a = 0;

int l=0;

int i;

for (i = 0; ; ++i)

{

a = pow(-1,i)\*((double)(pow(i,2)+1)/(double)(pow(i,3)+3));

l+=1;

if (l == k)

{

l = 0;

continue;

}

printf("%.4f\n", a);

n = n - 1;

if (n == 0)

break;

}

}

// while

#include <stdio.h>

#include <math.h>

void print(int n, int k)

{

double a = 0;

int l=0;

int i = 0;

while (1)

{

a = pow(-1,i)\*((double)(pow(i,2)+1)/(double)(pow(i,3)+3));

l+=1;

if (l == k)

{

l = 0;

++i;

continue;

}

printf("%.4f\n", a);

n = n - 1;

if (n == 0)

break;

++i;

}

}

// do while

#include <stdio.h>

#include <math.h>

void print(int n, int k)

{

double a = 0;

int l=0;

int i = 0;

do

{

a = pow(-1,i)\*((double)(pow(i,2)+1)/(double)(pow(i,3)+3));

l+=1;

if (l == k)

{

l = 0;

++i;

continue;

}

printf("%.4f\n", a);

n = n - 1;

if (n == 0)

break;

++i;

} while (1);

}

#### Задание 4

Написать функцию, которая возвращает номер первого члена последовательности , для которого выполняется условие .

Функция в качестве параметра принимает значение .

Выход из цикла организовать с помощью инструкции перехода break.

Прототип функции int findFirstElement(double eps).

Функция не должна использовать функции консольного ввода-вывода.

// for

#include <stdio.h>

#include <math.h>

int findFirstElement(double eps)

{

double a = 0;

int i;

for (i = 0;;++i)

{

a = pow(-1,i)\*((double)(pow(i,2)+1)/(double)(pow(i,3)+3));

if (fabs(a) <= eps)

{

return i+1;

break;

}

}

}

// while

#include <stdio.h>

#include <math.h>

int findFirstElement(double eps)

{

double a = 0;

int i = 0;

while (1)

{

a = pow(-1,i)\*((double)(pow(i,2)+1)/(double)(pow(i,3)+3));

if (fabs(a) <= eps)

{

return i+1;

break;

}

++i;

}

}

// do while

#include <math.h>

int findFirstElement(double eps)

{

double a = 0;

int i = 0;

do

{

a = pow(-1,i)\*((double)(pow(i,2)+1)/(double)(pow(i,3)+3));

if (fabs(a) <= eps)

{

return i+1;

break;

}

++i;

} while (1);

}

#### Задание 5

Написать функцию, которая возвращает номер первого отрицательного члена последовательности , для которого выполняется условие .

Функция в качестве параметра принимает значение .

Выход из цикла организовать с помощью инструкции перехода return.

Прототип функции int findFirstNegativeElement(double eps).

Функция не должна использовать функции консольного ввода-вывода.

// for

#include <stdio.h>

#include <math.h>

int findFirstNegativeElement(double eps)

{

double a = 0;

int i;

for (i = 0;;++i)

{

a = pow(-1,i)\*((double)(pow(i,2)+1)/(double)(pow(i,3)+3));

if ((a < 0) && (fabs(a) <= eps))

{

return i+1;

}

}

}

// while

#include <stdio.h>

#include <math.h>

int findFirstNegativeElement(double eps)

{

double a = 0;

int i = 0;

while (1)

{

a = pow(-1,i)\*((double)(pow(i,2)+1)/(double)(pow(i,3)+3));

if ((a < 0) && (fabs(a) <= eps))

{

return i+1;

}

++i;

}

}

// do while

#include <math.h>

int findFirstNegativeElement(double eps)

{

double a = 0;

int i = 0;

do

{

a = pow(-1,i)\*((double)(pow(i,2)+1)/(double)(pow(i,3)+3));

if ((a < 0) && (fabs(a) <= eps))

{

return i+1;

}

++i;

} while (1);

}

#### Задание 6

Разработать функцию main(), которая в цикле выводит на экран меню из 6 пунктов и ожидает ответа пользователя:

1. Задание 1
2. Задание 2
3. Задание 3
4. Задание 4
5. Задание 5
6. Выход

Обработку пользовательского ввода выполнить с помощью инструкции switch. При нажатии на соответствующую цифру меню стирается с экрана и приложение переходит в режим демонстрации работы выбранного задания. По завершении демонстрации программа ожидает пользовательского ввода для перехода в меню. Выход из программы должно осуществляться при нажатии цифры 6 в режиме меню. Скомпилировать 3 версии программы:

1. Все функции реализованы с помощью цикла for
2. Все функции реализованы с помощью цикла while
3. Все функции реализованы с помощью цикла do … while

Убедиться, что все три версии программы работают одинаково

// for

#include <stdio.h>

#include "task11.c"

#include "task12.c"

#include "task13.c"

#include "task14.c"

#include "task15.c"

int main(void)

{

int n;

int k;

int ir;

double eps;

double r;

for (;;)

{

system("cls");

printf("1 - Task11\n2 - Task12\n3 - Task13\n4 - Task14\n5 - Task15\n6 - Exit\r\n");

char answer = \_getch();

system("cls");

switch (answer)

{

case '1':

printf("\nEnter n:\n");

scanf("%i", &n);

printf("\n");

r = summ(n);

printf("%lf\n", r);

break;

case '2':

printf("\nEnter eps:\n");

scanf("%lf", &eps);

printf("\n");

r = summ2(eps);

printf("%lf\n", r);

break;

case '3':

printf("\nEnter n:\n");

scanf("%i", &n);

printf("\nEnter k:\n");

scanf("%i", &k);

printf("\n");

print(n, k);

break;

case '4':

printf("\nEnter eps:\n");

scanf("%lf", &eps);

printf("\n");

ir = findFirstElement(eps);

printf("%i\n", ir);

break;

case '5':

printf("\nEnter eps:\n");

scanf("%lf", &eps);

printf("\n");

ir = findFirstNegativeElement(eps);

printf("%i\n", ir);

break;

case '6':

return 0;

break;

default:

printf("Invalid input\n");

break;

}

system("pause");

}

return 0;

}

// for

print(n, k);

break;

case '4':

printf("\nEnter eps:\n");

scanf("%lf", &eps);

printf("\n");

ir = findFirstElement(eps);

printf("%i\n", ir);

break;

case '5':

printf("\nEnter eps:\n");

scanf("%lf", &eps);

printf("\n");

ir = findFirstNegativeElement(eps);

printf("%i\n", ir);

break;

case '6':

return 0;

break;

default:

printf("Invalid input\n");

break;

}

system("pause");

}

return 0;

}

// while

#include <stdio.h>

#include "Task21.c"

#include "Task22.c"

#include "Task23.c"

#include "Task24.c"

#include "Task25.c"

int main(void)

{

int n;

int k;

int ir;

double eps;

double r;

while (1)

{

system("cls");

printf("1 - Task21\n2 - Task22\n3 - Task23\n4 - Task24\n5 - Task25\n6 - Exit\r\n");

char answer = \_getch();

system("cls");

switch (answer)

{

case '1':

printf("\nEnter n:\n");

scanf("%i", &n);

printf("\n");

r = summ(n);

printf("%lf\n", r);

break;

case '2':

printf("\nEnter eps:\n");

scanf("%lf", &eps);

printf("\n");

r = summ2(eps);

printf("%lf\n", r);

break;

case '3':

printf("\nEnter n:\n");

scanf("%i", &n);

printf("\nEnter k:\n");

scanf("%i", &k);

printf("\n");

print(n, k);

break;

case '4':

printf("\nEnter eps:\n");

scanf("%lf", &eps);

printf("\n");

ir = findFirstElement(eps);

printf("%i\n", ir);

break;

case '5':

printf("\nEnter eps:\n");

scanf("%lf", &eps);

printf("\n");

ir = findFirstNegativeElement(eps);

printf("%i\n", ir);

break;

case '6':

return 0;

break;

default:

printf("Invalid input\n");

break;

}

system("pause");

}

return 0;

}

// while

print(n, k);

break;

case '4':

printf("\nEnter eps:\n");

scanf("%lf", &eps);

printf("\n");

ir = findFirstElement(eps);

printf("%i\n", ir);

break;

case '5':

printf("\nEnter eps:\n");

scanf("%lf", &eps);

printf("\n");

ir = findFirstNegativeElement(eps);

printf("%i\n", ir);

break;

case '6':

return 0;

break;

default:

printf("Invalid input\n");

break;

}

system("pause");

}

return 0;

}

// do while

#include <stdio.h>

#include "task31.c"

#include "task32.c"

#include "task33.c"

#include "task34.c"

#include "task35.c"

int main(void)

{

int n;

int k;

int ir;

double eps;

double r;

do

{

system("cls");

printf("1 - Task31\n2 - Task32\n3 - Task33\n4 - Task34\n5 - Task35\n6 - Exit\r\n");

char answer = \_getch();

system("cls");

switch (answer)

{

case '1':

printf("\nEnter n:\n");

scanf("%i", &n);

printf("\n");

r = summ(n);

printf("%lf\n", r);

break;

case '2':

printf("\nEnter eps:\n");

scanf("%lf", &eps);

printf("\n");

r = summ2(eps);

printf("%lf\n", r);

break;

case '3':

printf("\nEnter n:\n");

scanf("%i", &n);

printf("\nEnter k:\n");

scanf("%i", &k);

printf("\n");

print(n, k);

break;

case '4':

printf("\nEnter eps:\n");

scanf("%lf", &eps);

printf("\n");

ir = findFirstElement(eps);

printf("%i\n", ir);

break;

case '5':

printf("\nEnter eps:\n");

scanf("%lf", &eps);

printf("\n");

ir = findFirstNegativeElement(eps);

printf("%i\n", ir);

break;

case '6':

return 0;

break;

default:

printf("Invalid input\n");

break;

}

system("pause");

} while (1);

return 0;

}

// do while

print(n, k);

break;

case '4':

printf("\nEnter eps:\n");

scanf("%lf", &eps);

printf("\n");

ir = findFirstElement(eps);

printf("%i\n", ir);

break;

case '5':

printf("\nEnter eps:\n");

scanf("%lf", &eps);

printf("\n");

ir = findFirstNegativeElement(eps);

printf("%i\n", ir);

break;

case '6':

return 0;

break;

default:

printf("Invalid input\n");

break;

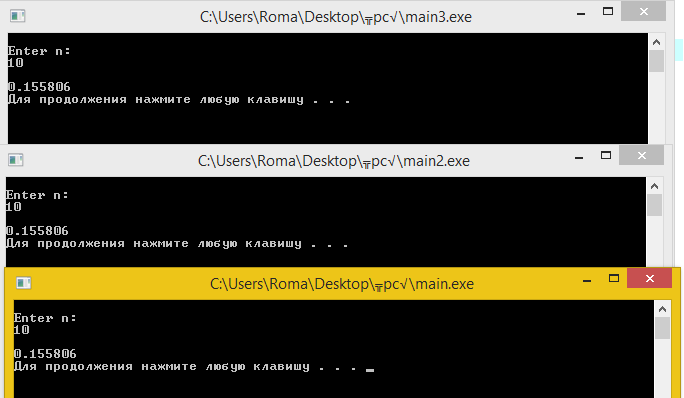
}

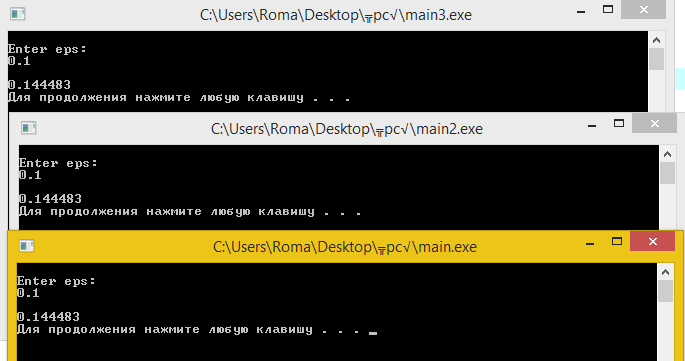
system("pause");

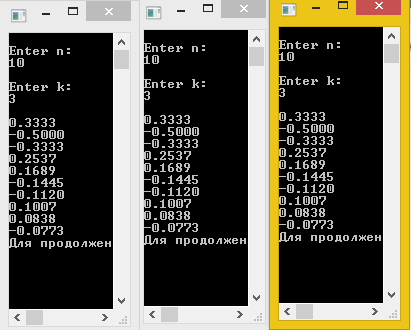
} while (1);

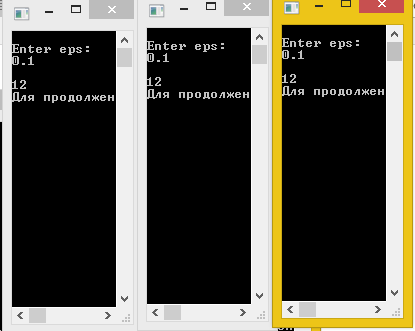
return 0;

}









## Практикум №4 Указатели и массивы

### Цель практикума

Овладеть навыками работы с указателями и массивами в языке программирвоания C

#### Задание 1

Линейный алгоритм функции из задания 1 практикума №1 разделить на две процедуры, выделив в одну вычислительные операции этого алгоритма, а в другую все операции ввода-вывода. Каждую процедуру оформить как функцию. Вычислительную часть алгоритма оформить как функцию с параметрами, передаваемыми по значению с использованием указателей на константные объекты, без возвращаемого значения. Результат вычисления вернуть через параметр-указатель. Прототип функции:

void f(const double \*x, double \*result) // Если функция имеет один параметр

void f(const double \*x, const double \*y, double \*result) // Если функция имеет два параметра

Другую часть алгоритма оформить как функцию void main(), вызывающую первую функцию нужное количество раз. Записать тексты функций файл с именем task1.с в следующем порядке: функция с параметрами, функция main. Cкомпилировать, скомпоновать и выполнить.

#include <stdio.h>

#include <math.h>

void f(const double \*x, double \*result)

{

\*result = sqrt(\*x);

}

#### Задание 2

Написать функцию вывода массива целых чисел на экран. Функция получает указатель на первый элемент массива и его длину.

Прототип функции void writeArray(int \*arr, int n).

Формат вывода: 1 2 3 4 5 6

#include <stdio.h>

#include <math.h>

void writeArray(int \*arr, int n)

{

int i;

for(i=0; i<n; i++)

{

printf("%d ", arr[i]);

}

}

#### Задание 3

Написать функцию ввода массива целых чисел с клавиатуры. Функция получает указатель на первый элемент массива и его длину.

Прототип функции void readArray(int \*arr, int n).

#include <stdio.h>

#include <math.h>

void readArray(int \*arr, int n)

{

int i;

printf("Input %d elements:\n", n);

for(i=0; i<n; i++)

{

scanf("%d", &arr[i]);

}

for(i=0; i<n; i++)

{

printf("%d ", arr[i]);

}

}

#### Задание 4

Написать функцию сортировки массива с использованием алгоритма сортировки, указанном в индивидуальном задании. Функция должна получать указатель на первый элемент массива и количество элементов в массиве.

Прототип функции void sort(int \*arr, int n).

Функция не должна использовать функции консольного ввода-вывода.

#include <stdio.h>

#include <math.h>

void sort(int \*arr, int n)

{

int i, j, step;

int tmp;

for (step = n / 2; step > 0; step /= 2)

{

for (i = step; i < n; i++)

{

tmp = arr[i];

for (j = i; j >= step; j -= step)

{

if (tmp < arr[j - step])

arr[j] = arr[j - step];

else

break;

}

arr[j] = tmp;

}

for(i=0; i<n; i++)

{

printf("%d ", arr[i]);

}

}

}

#### Задание 5

Разработать функцию main, демонстрирующую работу функций из заданий 1 и 4. Организовать меню и возможность многократной демонстрации заданий.

#include <stdio.h>

#include "Task41.c"

#include "Task42.c"

#include "Task43.c"

#include "Task44.c"

int main(void)

{

double x = 5;

double result;

const int n = 6;

int a[] = {1,2,3,4,5,6};

int ab[n];

int ac[]={4,3,6,2,1,5};

int ad[n];

do

{

system("cls");

printf("1 - Task41\n2 - Task42\n3 - Task43\n4 - Task44\n5 - Exit\r\n");

char answer = \_getch();

system("cls");

switch (answer)

{

case '1':

f(&x, &result);

printf("x = %.4lf\nf(x) = %.4lf", x, result);

printf("\nEnter x:\n");

scanf("%lf", &x);

f(&x, &result);

printf("x = %.4lf\nf(x) = %.4lf\n", x, result);

break;

case '2':

writeArray(a, n);

printf("\n");

break;

case '3':

readArray(ab, n);

printf("\n");

break;

case '4':

sort(ac, n);

printf("\n");

break;

case '5':

return 0;

break;

default:

printf("Invalid input\n");

break;

}

system("pause");

} while (1);

return 0;

}

printf("\n");

break;

case '4':

sort(ac, n);

printf("\n");

break;

case '5':

return 0;

break;

default:

printf("Invalid input\n");

break;

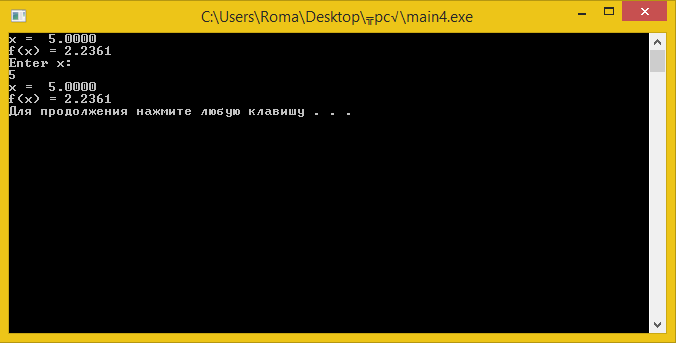
}

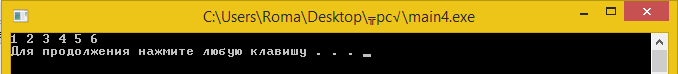
system("pause");

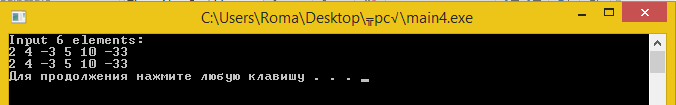
} while (1);

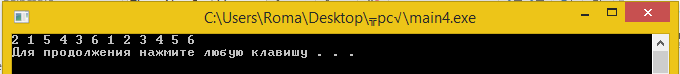
return 0;

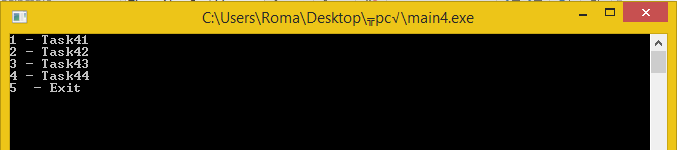
}











## Практикум №5 Строки

### Цель практикума

Овладеть навыками работы со строками в языке программирвоания C

#### Задание 1

Написать функцию подсчёта дины строки. Функция принимает исходную строку и возвращает количество символов в строке, не включая нулевой символ.

Прототип функции int strlen(const char \*str).

Функция не должна использовать функции консольного ввода-вывода.

int strlen(const char \*str)

{

int count = 0;

int i = 0;

while (str[i] != 0)

{

count++;

i++;

}

return count;

}

#### Задание 2

Написать функцию, поиска подстроки в строке. Функция принимает исходную строку и подстроку для поиска и возвращает адрес подстроки, если подстрока найдена, 0 в противном случае. Обращение к символам строки после нулевого символа недопустимо.

Прототип функции char \*find(const char \*str, const char \*substr).

Функция не должна использовать функции консольного ввода-вывода.

char \*find(const char \*str, const char \*substr)

{

int i = 0;

int j = 0;

int result = -1;

while (substr[j] != 0)

{

if (str[i] == 0)

return NULL;

if (str[i] == substr[j])

{

if (result == -1)

result = i;

i++;

j++;

}

else if (result != -1)

{

i = result + 1;

j = 0;

result = -1;

}

else

i++;

}

return str + result;

}

#### Задание 3

Написать функцию, удаления подстроки в строке. Функция принимает исходную строку и подстроку для поиска. С помощью функции find найти адрес начала подстроки и затем удалить эту подстроку. Оставшиеся в конце символы заполнить нулями. Обращение к символам строки после нулевого символа недопустимо.

Прототип функции void delete(char \*str, const char \*substr).

Функция не должна использовать функции консольного ввода-вывода.

void delete(const char \*str, const char \*substr)

{

int i = 0;

int j;

int substr\_length = strlen(substr);

char \*index = find(str, substr);

if (index == NULL)

return;

int f = 1;

for (i = 0; i < f; ++i)

{

++f;

index[i] = index[substr\_length + i];

if (index[substr\_length + i] == 0)

break;

}

for (j = 1; j < substr\_length; j++)

{

index[i + j] = 0;

}

}

Задание 4

Написать функцию копирования строк. Функция принимает исходную строку, буфер для копирования и размер буфера. Если длина исходной строки больше размера буфера, необходимо скопировать столько символов, сколько вместится в буфер, не забыв добавить в конце строки нулевой символ.

Прототип функции void strcpy(char \*dest, int len, const char \*src). Функция не должна использовать функции консольного ввода-вывода.

void strcpy(char \*dest, int len, const char \*src)

{

int i = 0;

for (i = 0; i < len - 1; i++)

{

dest[i] = src[i];

if (src[i] == 0)

return;

}

dest[i] = 0;

}

Задание 5

Написать функцию конкатенации строк. Функция принимает исходные строки, буфер для размещения и размер буфера. Если сумма длин исходных строк больше размера буфера, необходимо скопировать столько символов, сколько вместится в буфер, не забыв добавить в конце строки нулевой символ.

Прототип функции void strcat(char \*dest, int len, const char \*first, const char \*second). Функция не должна использовать функции консольного ввода-вывода.

void strcat(char \*dest, int len, const char \*first, const char \*second)

{

int i = 0;

int first\_length = strlen(first);

for (i = 0; i < len - 1; i++)

{

if (i < first\_length)

dest[i] = first[i];

else

{

dest[i] = second[i - first\_length];

if (second[i - first\_length] == 0)

return;

}

}

dest[i] = 0;

}

#### Задание 6

Написать функцию вставки подстроки в строку. Функция принимает исходнную строку, строку для вставки, индекс, определяющий место вставки, буфер для размещения новой строки и размер буфера. Если сумма длин исходной строки и строки для встивки больше размера буфера или индекс находится за пределами строки, необходимо вернуть соответствующую ошибку.

Прототип функции \_Bool insert(const char \*src, const char \*str, int index, char \*dest, int len).

Функция не должна использовать функции консольного ввода-вывода.

\_Bool insert(const char \*src, const char \*str, int index, char \*dest, int len)

{

int i = 0;

int j = 0;

if ((strlen(src) + strlen(str) + 1 > len) || (index < 0 || index > strlen(src) - 1))

return 0;

while (i < index)

{

dest[i] = src[i];

i++;

}

while (str[j] != 0)

{

dest[i] = str[j];

i++;

j++;

}

j = index;

while (src[j] != 0)

{

dest[i] = src[j];

i++;

j++;

}

dest[i] = 0;

return 1;

}

#### Задание 7

Написать функцию main, демонстрирующую работу ранее разработанных функций

void main()

{

char string[100];

char substring[100];

char buffstring[200];

printf("String: ");

gets(string);

printf("\n substring : ");

gets(substring);

printf("\n\n strlen function calculated initial string length");

printf("\n\t returned %d", strlen(string));

printf("\n\n find function searched for substring in the initial string");

printf("\n\t returned adress %x", find(string, substring));

if (find(string, substring) != NULL)

printf("\n\twhich is position #%d in initial string", find(string, substring) - string + 1);

printf("\n\nstrcpy function copied initial string in buffstring");

strcpy(buffstring, 200, string);

printf("\n\twith non-limiting buffer length value (200)");

printf("\n\t\tbuffstring content: %s", buffstring);

strcpy(buffstring, strlen(string) / 2 + 1, string);

printf("\n\twith limiting buffer length value (%d)", strlen(string) / 2 + 1);

printf("\n\t\tbuffstring content: %s", buffstring);

printf("\n\nstrcat function concatenated initial string & substring into buffstring");

strcat(buffstring, 200, string, substring);

printf("\n\twith non-limiting buffer length value (200)");

printf("\n\t\tbuffstring content: %s", buffstring);

strcat(buffstring, strlen(string) + strlen(substring) / 2, string, substring);

printf("\n\twith limiting buffer length value (%d)", strlen(string) + strlen(substring) / 2);

printf("\n\t\tbuffstring content: %s", buffstring);

printf("\n \n insert function tried to insert substring into initial string");

printf("\n \t using non-limiting buffer length value (200) & correct index (%d)", strlen(string) / 2);

printf("\n \t \t returned %s", insert(string, substring, strlen(string) / 2, buffstring, 200) ? "true" : "false");

printf("\n \t \t buffstring content: %s", buffstring);

printf("\n \t using limiting buffer length value (%d)", (strlen(string) + strlen(substring)) / 4 \* 3);

printf("\n \t\ treturned %s", insert(string, substring, strlen(string) / 2, buffstring, (strlen(string) + strlen(substring)) / 4 \* 3) ? "true" : "false");

printf("\n \t using incorrect index (%d)", strlen(string) + 5);

printf("\n \t \t returned %s", insert(string, substring, strlen(string) + 5, buffstring, 200) ? "true" : "false");

printf("\n \n delete function tried to remove substring from initial string");

delete(string, substring);

printf("\n \t string content: %s", string);

\_getch();

}

strlen(substring)) / 4 \* 3);

printf("\n \t\ treturned %s", insert(string, substring, strlen(string) / 2, buffstring, (strlen(string) + strlen(substring)) / 4 \* 3) ? "true" : "false");

printf("\n \t using incorrect index (%d)", strlen(string) + 5);

printf("\n \t \t returned %s", insert(string, substring, strlen(string) + 5, buffstring, 200) ? "true" : "false");

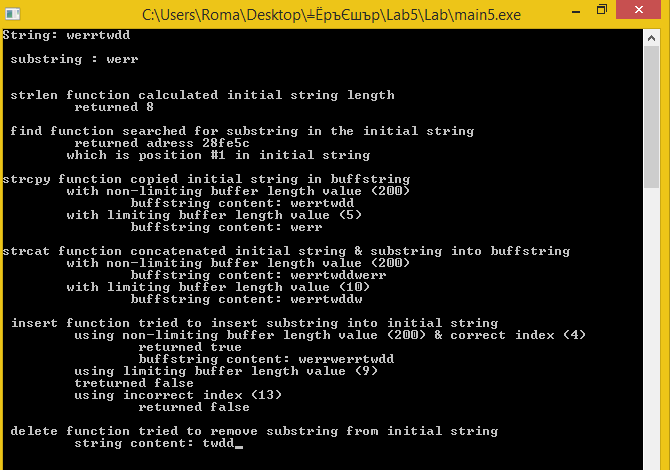
printf("\n \n delete function tried to remove substring from initial string");

delete(string, substring);

printf("\n \t string content: %s", string);

\_getch();

}



## Практикум №6 Структуры, объединения, перечисления и битовые поля

### Цель практикума

Овладеть навыками работы со структурами, объединениями, перечислениями и битовыми полями в языке C.

#### Задание 1

Разработать структуру «Зачётка». Структура включает в себя фамилию, имя, отчество студента, номер зачётной книжки, массив, хранящий информацию о каждом семестре. Размер массива равен количеству разворотов в зачётной книжке, предназначенных для проставления зачётов и экзаменов.

Информация о семестре включает в себя номер семестра, учебный год и массив, хранящий информацию о предметах. Размер массива равен количеству строк в Вашей зачётке (левая и правая сторона)

Информация о предмете включает в себя название предмета, тип оценки (зачет или экзамен), оценку, дату, фамилию преподавателя.

Оценка является объединением из двух полей: логическое поле (зачёт или незачёт) и перечисление, представляющее оценку.

Разработать консольное приложение, которое является электронной копией Вашей зачётки. Вся инфорация вводится статически (при желании можно вводить информацию с клавиатуры). Приложение должно показвать информацию об успеваемости студента по семестрам и при этом корретно различать экзамены и зачёты.

#include "stdio.h"

#include "locale.h"

enum MarkEnum { A = 5, B = 4, C = 3, D = 2};

struct BookList

{

char \*exam;

char \*hasPassed;

enum MarkEnum mark;

};

struct RecordBook

{

char\* lastName;

char\* firstName;

char\* middleName;

int number;

struct BookList bookList[3][2];

};

int main()

{

int i,j;

setlocale(LC\_ALL, "RUS");

struct RecordBook rb;

rb.lastName = "Çåëåíîõàò";

rb.firstName = "Ðîìàí";

rb.middleName = "Àëåêñàíäðîâè÷";

rb.number = 17061;

printf("Çà÷¸òíàÿ êíèæêà ¹%d\n", rb.number);

printf("ÔÈÎ ñòóäåíòà: %s %s %s\n\n", rb.lastName, rb.firstName, rb.middleName);

rb.bookList[0][0].exam = "Âû÷.Ìîä"; rb.bookList[0][0].hasPassed = "Ýêçàìåí"; rb.bookList[0][0].mark = A;

rb.bookList[0][1].exam = "Ïðàêòèêà"; rb.bookList[0][1].hasPassed = "Ýêçàìåí"; rb.bookList[0][1].mark = C;

rb.bookList[1][0].exam = "ÒÝÖ"; rb.bookList[1][0].hasPassed = "Ýêçàìåí"; rb.bookList[1][0].mark = B;

rb.bookList[1][1].exam = "ÎÒÑ"; rb.bookList[1][1].hasPassed = "Ýêçàìåí"; rb.bookList[1][1].mark = B;

rb.bookList[2][0].exam = "ÊÃ"; rb.bookList[2][0].hasPassed = "Ýêçàìåí"; rb.bookList[2][0].mark = C;

rb.bookList[2][1].exam = "ÑÔÈÔÒ"; rb.bookList[2][1].hasPassed = "Ïåðåñäà÷à"; rb.bookList[2][1].mark = D;

for (i = 0; i < 3; i++)

{

printf("%d ñåìåñòð:\n", i + 1);

for (j = 0; j < 2; j++)

{

printf("%s ", rb.bookList[i][j].exam);

printf("%s ", rb.bookList[i][j].hasPassed);

printf("%d ", rb.bookList[i][j].mark);

printf("\n");

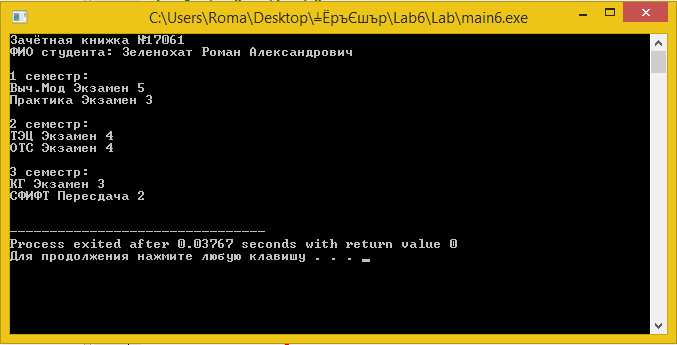
}

printf("\n");

}

return 0;

}



rb.bookList[2][0].exam = "ÊÃ"; rb.bookList[2][0].hasPassed = "Ýêçàìåí"; rb.bookList[2][0].mark = C;

rb.bookList[2][1].exam = "ÑÔÈÔÒ"; rb.bookList[2][1].hasPassed = "Ïåðåñäà÷à"; rb.bookList[2][1].mark = D;

for (i = 0; i < 3; i++)

{

printf("%d ñåìåñòð:\n", i + 1);

for (j = 0; j < 2; j++)

{

printf("%s ", rb.bookList[i][j].exam);

printf("%s ", rb.bookList[i][j].hasPassed);

printf("%d ", rb.bookList[i][j].mark);

printf("\n");

}

printf("\n");

}

return 0;

}

## Практикум №7 Динамическое выделение памяти

### Цель практикума

Овладеть навыками выделения динамической памяти в языке программирования C.

#### Задание 1

Написать функцию, которая создаёт в динамической памяти целое число, осуществляет ввод, выводит на экран адрес переменной и её значение, увеличивает значение на единицу и снова выводит адрес и значение на экран. Перед выходом из функции необходимо освободить ранее выделенную память.

Прототип функции void incrementHeapVariable().

Формат вывода: адрес значение адрес значение.

void incrementHeapVariable()

{

int\* numberPointer = (int\*)malloc(sizeof(int));

scanf\_s("%d", numberPointer);

printf("%x %d ", numberPointer, \*numberPointer);

\*numberPointer += 1;

printf("%x %d", numberPointer, \*numberPointer);

free(numberPointer);

}

#### Задание 2

Написать функцию, которая создаёт в динамической памяти из N элементов (число N вводится пользователем с клавиатуры), выводиn n на экран и с помощью разработанных функций (практикум №4, задания 2-4) осуществляет ввод данных, вывод массива на экран, сортировку и повторный вывод отсортированного массива на экран. Перед выходом из функции необходимо освободить выделенную ранее память.

Прототип функции void sortHeapArray(int \*arr, int n).

Формат вывода (первая строка – количество элементов массива):

5

5 1 9 7 8

1 5 7 8 9

void readArray(int\* arr, int n)

{

int i;

for (i = 0; i < n; i++)

scanf\_s("%d", arr + i);

}

void sort(int\* arr, int n)

{

int i;

int buff;

\_Bool unsorted;

do

{

unsorted = false;

for (i = 0; i < n - 1; i++)

if (arr[i] > arr[i + 1])

{

unsorted = true;

break;

}

if (unsorted)

{

for (i = 0; i < n - 1; i++)

if (arr[i] > arr[i + 1])

{

buff = arr[i];

arr[i] = arr[i + 1];

arr[i + 1] = buff;

}

}

else

break;

} while (unsorted);

}

void sortHeapArray(int\* arr, int n)

{

readArray(arr, n);

printf("%d\n", n);

writeArray(arr, n);

sort(arr, n);

writeArray(arr, n);

}

}

}

else

break;

} while (unsorted);

}

void sortHeapArray(int\* arr, int n)

{

readArray(arr, n);

printf("%d\n", n);

writeArray(arr, n);

sort(arr, n);

writeArray(arr, n);

}

#### Задание 3

Написать функцию, удаления подстроки в строке. Функция принимает исходную строку и подстроку для поиска. С помощью функции find найти адрес начала подстроки и затем удалить эту подстроку. Функция выделяет необходимое количество памяти в области динамической памяти, копирует необходимые части строки и возвращает указатель на вновь выделенную память. Обращение к символам строки после нулевого символа недопустимо.

Прототип функции char \*delete(char \*str, const char \*substr).

Функция не должна использовать функции консольного ввода-вывода.

int strlen(const char\* str)

{

int count = 0;

int i = 0;

while (str[i] != 0)

{

count++;

i++;

}

return count;

}

char\* find(const char\* str, const char\* substr)

{

int i = 0;

int j = 0;

int result = -1;

while (substr[j] != 0)

{

if (str[i] == 0)

return NULL;

if (str[i] == substr[j])

{

if (result == -1)

result = i;

i++;

j++;

}

else if (result != -1)

{

i = result + 1;

j = 0;

result = -1;

}

else

i++;

}

return str + result;

}

char\* delete(char\* str, const char\* substr)

{

int i = 0;

int j = 0;

int str\_length = strlen(str);

int substr\_length = strlen(substr);

char\* index = find(str, substr);

char\* newstr;

if (index == NULL)

return NULL;

newstr = (char\*)malloc(str\_length - substr\_length + 1);

for (;;)

{

if (str + i == index)

break;

newstr[i] = str[i];

i++;

}

for (;;)

{

newstr[i] = index[substr\_length + j];

if (index[substr\_length + i] == 0)

break;

i++;

j++;

}

return newstr;

}

{

i = result + 1;

j = 0;

result = -1;

}

else

i++;

}

return str + result;

}

char\* delete(char\* str, const char\* substr)

{

int i = 0;

int j = 0;

int str\_length = strlen(str);

int substr\_length = strlen(substr);

char\* index = find(str, substr);

char\* newstr;

if (index == NULL)

return NULL;

newstr = (char\*)malloc(str\_length - substr\_length + 1);

for (;;)

{

if (str + i == index)

break;

newstr[i] = str[i];

i++;

}

for (;;)

{

newstr[i] = index[substr\_length + j];

if (index[substr\_length + i] == 0)

break;

i++;

j++;

}

return newstr;

}

#### Задание 4

Написать функцию копирования строк. Функция принимает исходную строку, выделяет необходимое количество памяти в области динамической памяти, копирует содержимое исходной строки и возвращает указатель на вновь выделенную память

Прототип функции char \*strcpy(const char \*src).

Функция не должна использовать функции консольного ввода-вывода.

char\* strcpy(const char\* src)

{

int i = 0;

int len = strlen(src);

char\* newstr = (char\*)malloc(len \* sizeof(char) + 1);

for (i = 0; i < len; i++)

{

newstr[i] = src[i];

}

return newstr;

}

#### Задание 5

Написать функцию конкатенации строк. Функция принимает исходные строки, выделяет необходимое количество памяти в области динамической памяти, копирует содержимое строк и возвращает указатель на вновь выделенную память

Прототип функции char \*strcat(const char \*first, const char \*second).

Функция не должна использовать функции консольного ввода-вывода.

char\* strcat(const char\* first, const char\* second)

{

int i = 0;

int first\_length = strlen(first);

int second\_length = strlen(second);

char\* newstr = (char\*)malloc((first\_length + second\_length) \* sizeof(char) + 1);

for (i = 0; i < first\_length; i++)

{

newstr[i] = first[i];

}

for (i = 0; i < second\_length; i++)

{

newstr[first\_length + i] = second[i];

}

newstr[first\_length + i] = 0;

return newstr;

}

#### Задание 6

Написать функцию вставки подстроки в строку. Функция принимает исходнную строку, строку для вставки, индекс, определяющий место вставки, выделяет необходимое количество памяти в области динамической памяти, копирует реузльтат и возвращает указатель на вновь выделеную память.

Прототип функции char \*insert(const char \*src, const char \*str, int index).

Функция не должна использовать функции консольного ввода-вывода.

char\* insert(const char\* src, const char\* str, int index)

{

int i = 0;

int j = 0;

char\* newstr = (char\*)malloc((strlen(src) + strlen(str)) \* sizeof(char) + 1);

while (i < index)

{

newstr[i] = src[i];

i++;

}

while (str[j] != 0)

{

newstr[i] = str[j];

i++;

j++;

}

j = index;

while (src[j] != 0)

{

newstr[i] = src[j];

i++;

j++;

}

newstr[i] = 0;

return newstr;

}

int main()

{

int N;

int\* arr;

printf("razmer stroki : ");

scanf\_s("%d", &N);

arr = (int\*)malloc(N \* sizeof(int));

sortHeapArray(arr, N);

free(arr);

incrementHeapVariable();

\_getch();

}

