Министерство цифрового развития, связи и массовых коммуникаций

Российской Федерации

Ордена Трудового Красного Знамени

федеральное государственное бюджетное образовательное учреждение

высшего образования

Московский технический университет связи и информатики

(МТУСИ)

Кафедра «Информационная безопасность»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

по дисциплине

««Программирование в системах информационной безопасности»

на тему

«Компиляция файлов исходного кода и компоновка полученных объектных файлов в исполняемый модуль. Линейные алгоритмы»

Вариант №19

Выполнил:

студент группы БСУ1801 Плотников П.С

Проверил:

Старший преподаватель кафедры ИБ

Барков В. В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва 2021

# **Цель**

Овладеть навыками создания однофайловых и многофайловых проектов в интегрированной среде разработки Microsoft Visual Studio 2019 Community Edition и научится создавать линейные программы на языке C с применением арифметических операций.

# **Индивидуальные варианты заданий**

# **Результаты**

Для исходных данных, значения которых задать в виде констант в тексте main:

Для исходных произвольных данных, значения которых задать в виде вводит с клавиатуры в процессе выполнения программы:

**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.

#include <stdio.h>

#include <math.h>

void main(void)

{

double x = 1;

double f = pow(((1 + x + pow(x, 2)) / (2 \* x + pow(x, 2)) + 2 - (1 - x - pow(x, 2)) / (2 \* x - pow(x, 2))), -1) \* (5 - 2 \* pow(x, 2));

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

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

printf("x = ");

scanf("%lf", &x);

f = pow(((1 + x + pow(x, 2)) / (2 \* x + pow(x, 2)) + 2 - (1 - x - pow(x, 2)) / (2 \* x - pow(x, 2))), -1) \* (5 - 2 \* pow(x, 2));

printf("f = %.4lf", f);

}

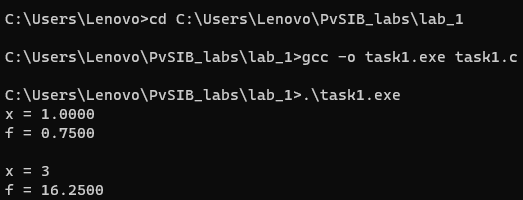


Рисунок 1 – Результат компиляции, компоновки и выполнения task1.exe

**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.

Листинг 2 – Исходный код файла task2.c

#include <stdio.h>

#include <math.h>

double f(double x)

{

return pow(((1 + x + pow(x, 2)) / (2 \* x + pow(x, 2)) + 2 - (1 - x - pow(x, 2)) / (2 \* x - pow(x, 2))), -1) \* (5 - 2 \* pow(x, 2));

}

void main(void)

{

double x = 9;

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

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

printf("x = ");

scanf("%lf", &x);

printf("f = %.4lf", f(x));

}

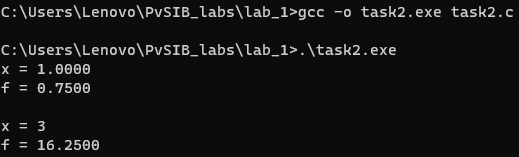


Рисунок 2 – Результат компиляции, компоновки и выполнения task2.exe

**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.

Листинг 3 – Исходный код файла task3.c

#include <stdio.h>

#include <math.h>

double f(double x);

void main(void)

{

double x = 1;

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

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

printf("x = ");

scanf("%lf", &x);

printf("f = %.4lf", f(x));

}

double f(double x)

{

return pow(((1 + x + pow(x, 2)) / (2 \* x + pow(x, 2)) + 2 - (1 - x - pow(x, 2)) / (2 \* x - pow(x, 2))), -1) \* (5 - 2 \* pow(x, 2));

}

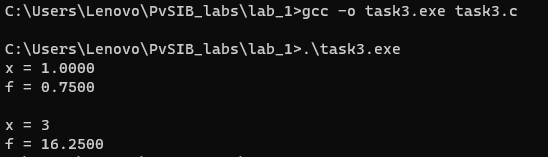


Рисунок 3 – Результат компиляции, компоновки и выполнения task3.exe

**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.

Листинг 4 – Исходный код файла task4.c

#include <stdio.h>

#include <math.h>

double x, result;

void f(void);

void main (void)

{

x = 1;

f();

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

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

printf("x =");

scanf("%lf", &x);

f();

printf("f = %.4lf", result);

}

void f(void)

{

result = pow(((1 + x + pow(x, 2)) / (2 \* x + pow(x, 2)) + 2 - (1 - x - pow(x, 2)) / (2 \* x - pow(x, 2))), -1) \* (5 - 2 \* pow(x, 2));

}

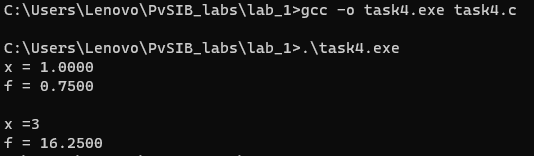


Рисунок 4 – Результат компиляции, компоновки и выполнения task4.exe

**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.

Листинг 5 – Исходный код файла task5\_main.c

#include <stdio.h>

double f(double x);

void main(void)

{

double x = 1;

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

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

printf("x = ");

scanf("%lf", &x);

printf("f = %.4lf", f(x));

}

Листинг 6 – Исходный код файла task5\_func.c

#include <math.h>

double f(double x)

{

return pow(((1 + x + pow(x, 2)) / (2 \* x + pow(x, 2)) + 2 - (1 - x - pow(x, 2)) / (2 \* x - pow(x, 2))), -1) \* (5 - 2 \* pow(x, 2));;

}

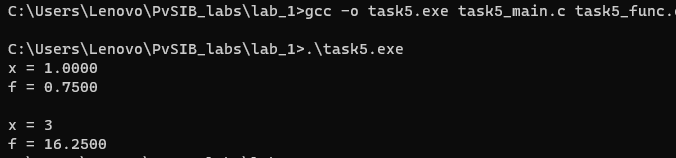


Рисунок 5 – Результат компиляции, компоновки и выполнения task5.exe

**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.

Листинг 7 – Исходный код файла task6\_main.c

#include <stdio.h>

extern double x, result;

void f(void);

void main (void)

{

x = 1;

f();

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

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

printf("x =");

scanf("%lf", &x);

f();

printf("f = %.4lf", result);

}

Листинг 8 – Исходный код файла task6\_func.c

#include <math.h>

double x, result;

void f(void)

{

result = pow(((1 + x + pow(x, 2)) / (2 \* x + pow(x, 2)) + 2 - (1 - x - pow(x, 2)) / (2 \* x - pow(x, 2))), -1) \* (5 - 2 \* pow(x, 2));

}

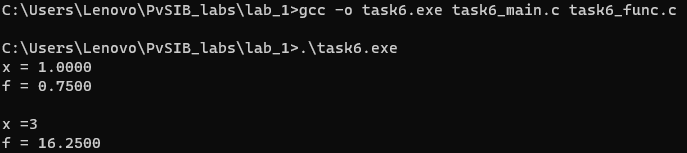


Рисунок 6 – Результат компиляции, компоновки и выполнения task6.exe

**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.

Листинг 9 – Исходный код файла task7\_main.c

#include <stdio.h>

#include "func.h"

extern double x,result;

void main (void)

{

x = 1;

f();

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

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

printf("x =");

scanf("%lf", &x);

f();

printf("f = %.4lf", result);

}

Листинг 10 – Исходный код файла task7\_func.c

#include <math.h>

#include "func.h"

double x,result;

void f(void)

{

result = pow(((1 + x + pow(x, 2)) / (2 \* x + pow(x, 2)) + 2 - (1 - x - pow(x, 2)) / (2 \* x - pow(x, 2))), -1) \* (5 - 2 \* pow(x, 2));

}

Листинг 11 – Исходный код файла func.h

#ifndef FUNC\_H

#define FUNC\_H

void f(void);

#endif

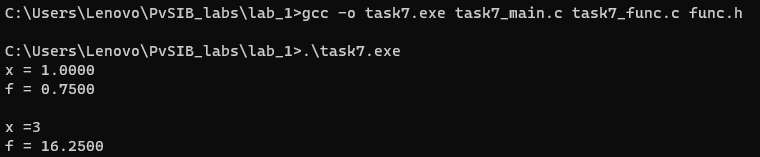


Рисунок 7 – Результат компиляции, компоновки и выполнения task7.exe

**Задание 8**

Скомпилировать по отдельности файлы из задания 7 task7\_main. obj и task7\_func. obj. 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.

Компонуем task7\_func.c и func.h в файл библиотеки task8\_lib.lib. Создаем консольное приложение с task8\_lib.lib и task7\_main.c, компилируем и компонуем.

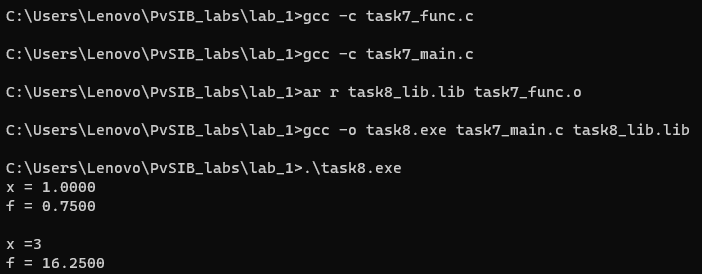


Рисунок 8 - Результат компиляции, компоновки и выполнения task8.exe

# [**https://github.com/Pavel-Plotnikov-BSU1801/Laboratornai-1-Plotnikov-P.S.-BSU1801.git**](https://github.com/Pavel-Plotnikov-BSU1801/Laboratornai-1-Plotnikov-P.S.-BSU1801.git)

# **Вывод**

В данной работе были разобраны процесс создания .c файлов, компиляции их в .obj файлы и процесс их компоновки в .exe файлы. Так же рассмотрели особенности работы с заголовочными файлами, объявление глобальных переменных и различные типы переменных и функций.