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 24

Выполнил:

студент группы БСУ1801

Спирин В.В.

Проверил:

старший преподаватель кафедры ИБ

Барков В.В.

Москва, 2021

# Цель работы

Овладеть навыками создания однофайловых и многофайловых проектов в интегрированной среде разработки Microsoft Visual Studio 2015 Community Edition 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.

**Индивидуальное задание**

24 вариант

**Общее Задание**

***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.

***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.

***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.

***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.

***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.

***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.

***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.

***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.

# Выполнение

Листинг 1 – программный код файла task1.c

#include <stdio.h>

#include<math.h>

void main(void)

{

double x = 4; //определение x

double f; //определение f

f = -sqrt(x); //вычисление f

printf("x = %.4lf\n", x); //вывод x

printf("f = %.4lf\n", f); //вывод f с 4 знаками после запятой

printf("x = ");

scanf\_s("%lf", &x); //ввод x с клавиатуры

f = -sqrt(x);

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

system("pause"); //оставляет окно открытым после выполнения программы

}

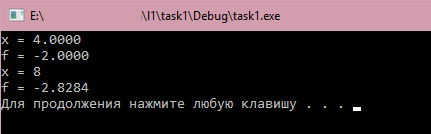


Рисунок 1 – Результат выполнения task1.exe

Листинг 2 – программный код файла task2.cpp

#include <stdio.h>

#include<math.h>

double f(double x) //определение функции для вычисления f

{

return -sqrt(x); //возврат значения функции

}

void main(void)

{

double x = 16;

double fres;

fres = f(x); //вызов функции f(x) и сохранение значения

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

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

printf("x = ");

scanf\_s("%lf", &x);

fres = f(x);

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

system("pause");

}

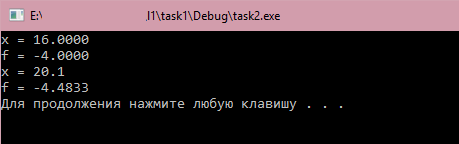


Рисунок 2 – Результат выполнения task2.exe

Листинг 3 – программный код файла task3.c

#include <stdio.h>

#include<math.h>

double f(double x); //объявление (прототип) функции f

void main(void)

{

double x = 64;

double fres;

fres = f(x);

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

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

printf("x = ");

scanf\_s("%lf", &x);

fres = f(x);

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

system("pause");

}

double f(double x)

{

return -sqrt(x);

}

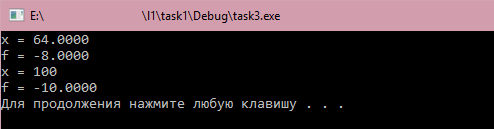


Рисунок 3 – Результат выполнения task3.exe

Листинг 4 – программный код файла task4.c

#include <stdio.h>

#include <math.h>

void f(void);

double x = 128;

double result;

void main(void)

{

f(); //вызов функции для вычисления f

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

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

printf("x = ");

scanf\_s("%lf", &x);

f();

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

system("pause");

}

void f(void)

{

result = -sqrt(x); // в переменную result записывается результат вычисления

}

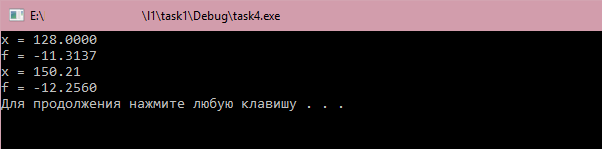


Рисунок 4 – Результат выполнения task4.exe

Листинг 5 – программный код файла task5\_main.c

#include <stdio.h>

double f(double x);

void main(void)

{

double x = 64;

double fres;

fres = f(x);

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

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

printf("x = ");

scanf\_s("%lf", &x);

fres = f(x);

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

system("pause");

}

Листинг 6 – программный код файла task5\_func.c

#include <math.h>

double f(double x)

{

return -sqrt(x);

}

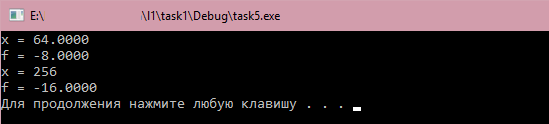


Рисунок 5 – Результат выполнения task5.exe

Вывод: несмотря на то что определение функции и вызов функции расположены и скомпилированы в разных файлах, после компоновки функция стала доступна.

Листинг 7 – программный код файла task6\_main.c

#include <stdio.h>

void f(void);

extern double x, result; //объявление переменных с внешним связыванием

void main(void)

{

f();

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

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

printf("x = ");

scanf\_s("%lf", &x);

f();

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

system("pause");

}

Листинг 8 – программный код файла task6\_func.c

#include <math.h>

double x = 512;

double result;

void f(void)

{

result = -sqrt(x);

}

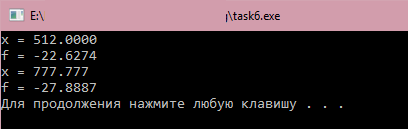


Рисунок 6– Результат выполнения task6.exe

Вывод: для доступа к переменным из других единиц трансляции их необходимо объявить с внешним связыванием extern. Глобальные переменные по умолчанию имеют внешнее связывание

Листинг 9 – программный код файла task7\_main.c

#include <stdio.h>

#include "func.h"

void main(void)

{

f();

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

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

printf("x = ");

scanf\_s("%lf", &x);

f();

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

system("pause");

}

Листинг 10 – программный код файла task7\_func.c

#include <math.h>

double x = 1024;

double result;

void f(void)

{

result = -sqrt(x);

}

Листинг 11 – программный код файла func.h

extern double x, result;

void f(void);

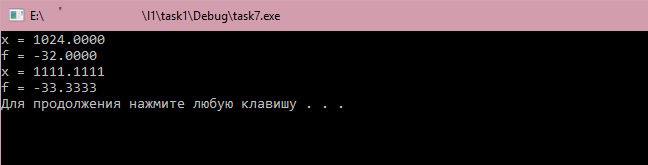
****

Рисунок 7 – Результат выполнения task7.exe

Листинг 12 – Команды командной строки Cross Tools для VS

cl -c task7\_func.c task7\_main.c

lib /OUT:task8\_lib.lib task7\_func.obj

link Task7\_main.obj task8\_lib.lib

Task7\_main.exe

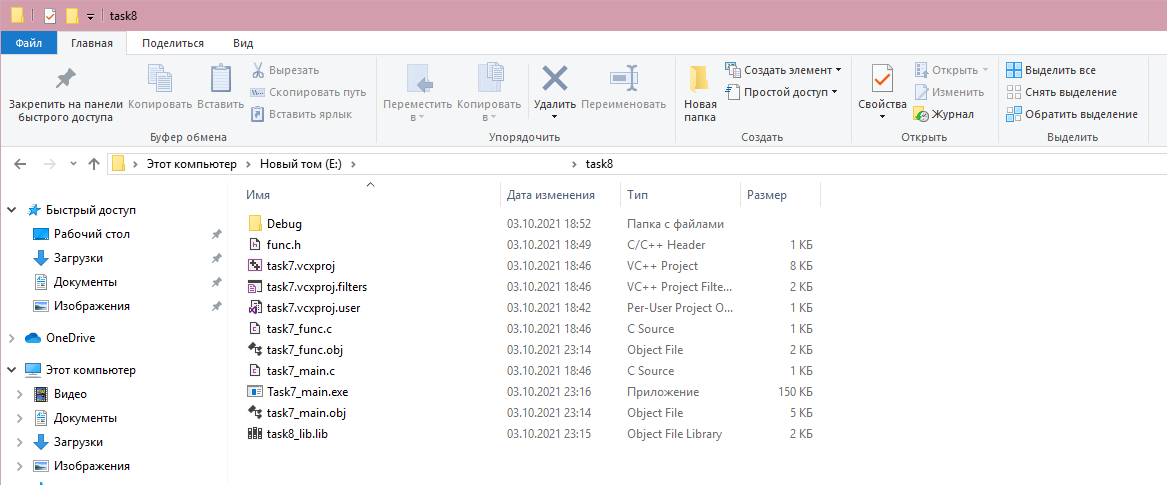
****

Рисунок 8 – Выполнение задания 8

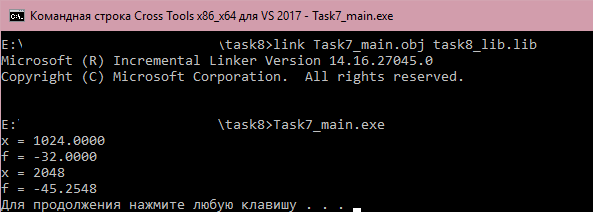


Рисунок 9 – Выполнение задания 8