Министерство Цифрового Развития, Связи и Массовых Коммуникаций Российской Федерации  
Федеральное Государственное Бюджетное Образовательное Учреждение Высшего Образования Ордена Трудового Красного знамени   
«Московский технический университет связи и информатики»

Кафедра информационной безопасности

Лабораторная работа №1

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

Выполнил  
Студент группы БПЗ1801

Шишкин Иван Ильич

Проверил:

Барков Вячеслав Валерьевич

Москва 2021

**ОБЩЕЕ ЗАДАНИЕ.**

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

**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.

**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.

**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)).

**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.

**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.

**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.

**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.

**ИНДИВИДУАЛЬНЫЙ ВАРИАНТ ЗАДАНИЯ.**

Вариант 24.

**ПРОГРАММНЫЙ КОД.**

**Задание 1.**

Листинг 1 – Исходный код файла task1.cpp

#include <iostream>

using namespace std;

int main()

{

int x = 16;

double f;

cout << "Const = " << x;

f = -1 \* sqrt(x);

cout << "\nIf x = " << x << " f = " << f;

cout << "\nEnter x = ";

cin >> x;

if (x < 0)

{

cout << "If x = " << x << " Error, x < 0" << "\n";

}

else

{

f = -1 \* sqrt(x);

cout << "If x = " << x << " f = " << f << "\n";

}

}

**Задание 2.**

Листинг 2 – Исходный код файла task2.cpp

#include <iostream>

using namespace std;

double Function(double count)

{

return -1 \* sqrt(count);

}

void main()

{

int x = 16;

double f;

cout << "Constant = " << x;

f = Function(x);

cout << "\nIf x = " << x << ", Function = " << f << "\nEnter x = ";

cin >> x;

if (x < 0)

{

cout << "If x = " << x << ", Error x<0\n" ;

}

else

{

f = Function(x);

cout << "If x = " << x << ", Function = " << f << "\n";

}

}

**Задание 3.**

Листинг 3 – Исходный код файла task3.cpp

#include <iostream>

using namespace std;

double Function(double count);

void main()

{

int x = 16;

double f;

cout << "Constant = " << x;

f = Function(x);

cout << "\nIf x = " << x << ", Function = " << f << "\nEnter x = ";

cin >> x;

if (x < 0)

{

cout << "If x = " << x << ", Error x<0";

}

else

{

f = Function(x);

cout << "If x = " << x << ", Function = " << f << "\n";

}

}

double Function(double count)

{

return -1 \* sqrt(count);

}

**Задание 4.**

Листинг 4 – Исходный код файла task4.cpp

#include <iostream>

using namespace std;

int x = 16;

double f;

void Function()

{

f = -1 \* sqrt(x);

}

void main()

{

cout << "Constant = " << x;

Function();

cout << "\nIf x = " << x << ", Function = " << f << "\nEnter x = ";

cin >> x;

if (x < 0)

{

cout << "\nIf x = " << x << ", Error x<0\n";

}

else

{

Function();

cout << "If x = " << x << ", Function = " << f << "\n";

}

}

**Задание 5.**

Листинг 5 – Исходный код файла task5.cpp

#include <iostream>

using namespace std;

double Function(double count);

void main()

{

int x = 16;

double f;

cout << "Constant = " << x;

f = Function(x);

cout << "\nIf x = " << x << ", Function = " << f << "\nEnter x = ";

cin >> x;

if (x < 0)

{

cout << "If x = " << x << ", Error x<0\n";

}

else

{

f = Function(x);

cout << "If x = " << x << ", Function = " << f << "\n";

}

}

Листинг 6 – Исходный код файла task5\_function.cpp

#include <math.h>

double Function(double count)

{

return -1 \* sqrt(count);

}

**Задание 6.**

Листинг 7 – Исходный код файла task6.cpp

#include <iostream>

using namespace std;

void Function();

extern double x, f;

void main()

{

cout << "Constant = " << x;

Function();

cout << "If x = " << x << ", Function = " << f << "\nEnter x = ";

cin >> x;

if (x < 0)

{

cout << "If x = " << x << ", Error x<0\n";

}

else

{

Function();

cout << "If x = " << x << ", Function = " << f << "\n";

}

}

Листинг 8 – Исходный код файла task6\_function.cpp

#include <math.h>

double x;

double f;

void Function()

{

f = -1 \* sqrt(x);

}

**Задание 7.**

Листинг 9 – Исходный код файла task7.cpp

#include <iostream>

#include "func.h"

using namespace std;

void main()

{

cout << "Constant = " << x;

Function();

cout << "\nIf x = " << x << ", Function = " << result << "\nEnter x = ";

cin >> x;

Function();

if (x < 0)

{

cout << "If x = " << x << ", Error x<0\n";

}

else

{

cout << "If x = " << x << ", Function = " << result << "\n";

}

}

Листинг 10 – Исходный код файла task7\_function.cpp

#include <math.h>

double result;

double x = 16;

void Function()

{

result = -1 \* sqrt(x);

}

Листинг 11 – Исходный код файла func.h

extern double x, result;

void Function();

**Задание 8.**

Листинг 12 – включение в проект заголовочного файла func.h и объектного файла task7\_func.obj для сборки библиотеки

<ItemGroup>

    <ClInclude Include="..\task7\func.h" />

  </ItemGroup>

  <ItemGroup>

    <Object Include="..\task7\x64\Debug\task7\_function.obj" />

  </ItemGroup>

Листинг 13 – включение в проект библиотеки task8\_lib.lib и объектного файла task7.obj для сборки исполняемого файла

  <ItemGroup>

    <Library Include="..\x64\Debug\task8\_lib.lib" />

  </ItemGroup>

  <ItemGroup>

    <Object Include="..\task7\x64\Debug\task7.obj" />

  </ItemGroup>

**СКРИНШОТЫ.**

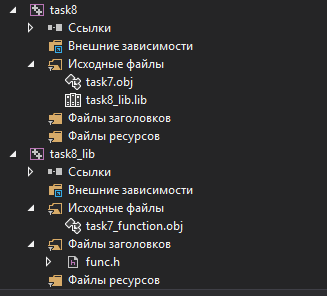


Рисунок 1 – Файлы проектов task8, task8\_lib.



Рисунок 2 – Результат работы программы.

**ВЫВОДЫ.**

В данной лабораторной работе мною были изучены: метод создания линейных программ с использованием дополнительных библиотек и функций, различные типы функций, особенности глобальных и локальных переменных и области их применения, метод создания статической библиотеки.