# Лабораторная работа №10. Функции во внешних файлах

## Цель и порядок работы

Цель работы – изучить возможности языка по созданию пользовательских функций, расположенных в отдельных файлах.

Порядок выполнения работы:

* ознакомиться с описанием лабораторной работы;
* получить задание у преподавателя, согласно своему варианту;
* написать программу и отладить ее на ЭВМ;
* оформить отчет.

## Краткая теория

В C++ используются различные виды функций. Прежде всего это стандартные и библиотечные функции. Стандартные функции входят в стандартную библиотеку, поставляемую вместе с компилятором. К библиотечным относятся любые функции, описание и реализация которых находятся в библиотеках, не входящих в стандартный набор.

Библиотеки и заголовочные файлы подключаются при помощи директивы препроцессора #include. После директивы препроцессора указывается имя библиотеки или имя заголовочного файла библиотеки. Имя заключается в угловые скобки (для стандартных библиотек) или в двойные кавычки (для остальных библиотек).

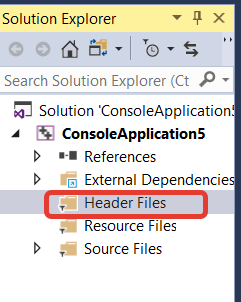
Например, для того чтобы подключить библиотеку математических функций необходимо в начале файла с исходным кодом указать следующую директиву:

#include <math.h>

А если библиотека располагается в том же каталоге, что и приложение (например библиотека mylib), то подключение библиотеки будет выглядет следующим образом:

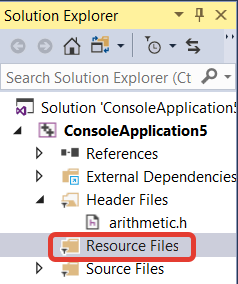
#include "mylib.h"

В угловых скобках или в двойных кавычках указывается имя специального заголовочного файла (расширение .h образовано от английского слова header – заголовок). В заголовочных файлах приводятся прототипы функций, а реализация, обычно, размещается в файле с таким же именем, но расширение меняется на .cpp

**Пример:**

Создаем в проекте заголовочный файл **arithmetic.h**. Для этого в обозревателе решений нужно вызвать контекстное меню в разделе **Header Files** и выбратьпункт **Add** – **New Item**, в открывшемся окне выбираем элемент **header file (.h)** и устанавливаем имя файла **arithmetic.h**. Внутри созданного файла запишем прототипы будущих функций add и sub.

double add(double a, double b);// прототип функции add

double sub(double a, double b);// прототип функции sub

Программный код самих функций должен располагаться в файле **arithmetic.cpp**. Данный файл должен располагаться в разделе **Resource File**.Внутри файла разместим следующий код

#include"arithmetic.h"

using namespace std;

double add(double a, double b)

{

return a + b;

}

double sub(double a, double b)

{

return a - b;

}

В тексте файла **.cpp**, расположенного в разделе **Source Files** (в этом файле находится функция **main()**), напишем программу вызывающую функции **add**() и **sub**(). Для того что бы обращение к данным функциям стало возможным, в начале файла добавим директиву #include, подключающую заголовочный файл "arithmetic.h".

#include <iostream>

#include"arithmetic.h"

using namespace std;

int main()

{

cout << add(10, 20) << endl;

cout << sub(15, 8) << endl;

system("pause");

return 0;

}

**Стандартные библиотеки**

Каждая стандартная библиотека имеет свой собственный заголовочный файл, содержащий прототипы всех функций данной библиотеки, а также определения различных констант, типов данных и т.д., необходимых функциям.

Список некоторых стандартных библиотек функций

|  |  |
| --- | --- |
| **Заголовочный файл** | **Содержимое заголовочного файла** |
| <ctype.h> | Содержит прототипы функций, проверяющих различные характеристики символов, а также позволяющих выполнять преобразования символов. |
| <errno.h> | Содержит макросы, полезные для сообщений об ошибках. |
| <float.h> | Содержит пределы значений для чисел с плавающей точкой в системе |
| <limits.h> | Содержит пределы значений для целочисленных чисел в системе |
| <locale.h> | Содержит прототипы функций и другую информацию, которая позволяет изменять работу программы и параметры окружения в соответствии с указанными локальными (в смысле региона, страны) настройками: кодировка символов, представление чисел и т.д. |
| <math.h> | Содержит прототипы математических функций. |
| <signal.h> | Содержит прототипы функций и макросы для обработки различных условий, которые могут возникать во время работы программы. |
| <stdarg.h> | Определяет макросы обработки списка параметров функции, для которой неизвестно число параметров и их тип. |
| <stddef.h> | Содержит общие определения типов, используемых Си для выполнения некоторых вычислений. |
| <stdio.h> | Содержит прототипы функций ввода/вывода и информацию, используемую ими. |
| <stdlib.h> | Содержит прототипы функций преобразования чисел в текст и текста в число, прототипы функций размещения памяти, генерации случайных чисел и других сервисных функций. |
| <string.h> | Содержит прототипы функций обработки строк. |
| <time.h> | Содержит прототипы функций и типы данных для функций управления временем и датой. |

**Библиотеки пользователя и заголовочные файлы**

Обычно все библиотеки располагаются в каталогах, прописанных в служебных переменных LIB или INCLUDE или в одном каталоге с основной программой. Однако, если библиотека располагается в некотором специфическом месте, то вместе с именем следует указать полный путь к ней. Следует отметить, что такая организация является не корректной, поскольку привязывается к текущему состоянию файловой системы. Случайное удаление файла библиотеки или перенос исходных текстов на другую машину или даже просто в другой каталог могут привести к невозможности сборки такой программы.

**Создание статической библиотеки (LIB-файла)**

Статические библиотеки являются хорошим инструментом повторного использования. Вы записываете в статическую библиотеку какой-либо код и затем обращаетесь к нему из приложений, причем данное обращение не требует добавления каких-либо файлов к проекту.

**Пример создания статической библиотеки**

1. Создайте новый проект, при создании выберите тип проекта **Static Library**, укажите проекту имя **Staticlibrary**.
2. добавьте в проект заголовочный файл ***Staticlib.h***.
3. Добавьте в заголовочный файл объявление функций:

// Staticlib.h

double Add(double a, double b);

// Returns a - b

double Subtract(double a, double b);

// Returns a \* b

double Multiply(double a, double b);

// Returns a / b

double Divide(double a, double b);

1. Создайте файл для кода библиотеки ***Staticlib.cpp***.
2. Внутри файла ***Staticlib.cpp***разместите код реализации математических функций:

// Staticlib.cpp

#include "pch.h"

#include "Staticlib1.h"

#include <stdexcept>

using namespace std;

double Add(double a, double b)

{

return a + b;

}

double Subtract(double a, double b)

{

return a - b;

}

double Multiply(double a, double b)

{

return a \* b;

}

double Divide(double a, double b)

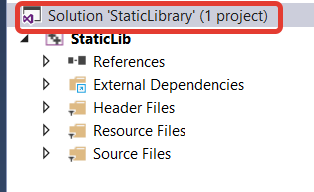
{

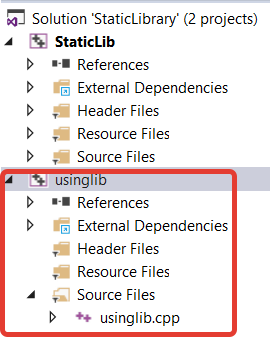
return a / b;

}

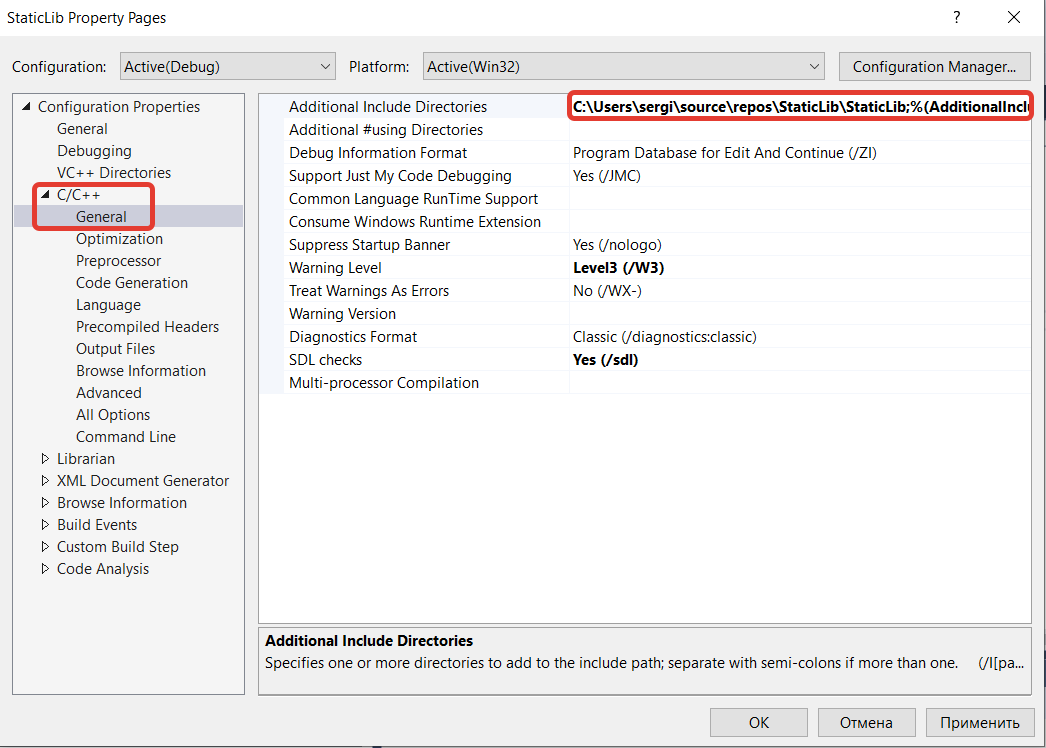
1. Скомпилируйте статическую библиотеку, выбрав **Build- Build Solution** в строке меню. В ходе компиляции будет создана статическая библиотека, которая может использоваться другими программами. (в папке Debug должен появиться файл с расширением .lib).

**Создание консольного приложения, использующего библиотеку**

1. В Обозревателе решений вызовите контекстное меню на верхнем элементе **StaticLibrary.** В открывшемся меню выберите **Add** - **New Project**. Выберите консольное приложение и укажите имя **usinglib**. В обозреватели решений должен появиться проект с указанным именем



1. Для использования математических процедур из статической библиотеки необходимо сослаться на эту библиотеку. В **обозревателе решений** в приложении **usinglib** вызовите контекстное меню на элементе **References**, в открывшемся меню выберите **Add** **reference**.
2. В открывшемся окне перейдите на вкладку **Project** и установите переключатель напротив имени библиотеки, нажмите **ОК** .
3. Укажем путь к файлу staticlib.h. Для этого в диалоговом окне **Properties** проекта **Usinglib** перейдите в раздел **C/C++ –** **General**. В поле **Additional include directories**, укажите путь к папке, в которой находится заголовочный файл библиотеки. Сохраните изменения.



1. Теперь, после подключения заголовочного файла в “**staticlib.h**” в коде проекта вы можете использовать библиотечные функции в этом приложении. Замените содержимое файла Usinglib.cpp следующим кодом:

// Usinglib.cpp

#include <iostream>

#include "Staticlib1.h"

using namespace std;

int main()

{

double a = 7.4;

double b = 99.0;

cout << "a + b = " <<

Add(a, b) << endl;

cout << "a - b = " <<

Subtract(a, b) << endl;

cout << "a \* b = " <<

Multiply(a, b) << endl;

cout << "a / b = " <<

Divide(a, b) << endl;

return 0;

}

1. Соберите исполняемый файл, выбрав **Build** – **Build Solution** в строке меню.

**Запуск приложения**

1. Убедитесь, что проект **Usinglib** выбран в качестве проекта по умолчанию, открыв контекстное меню для **Usinglib** в **обозревателе решений** и выбрав пункт **Set As StartUp Project**.
2. Чтобы запустить проект, в строке меню выберите **Debug** – **Start Without Debugging**. Выходные данные должны иметь:

a + b = 106.4

a - b = -91.6

a \* b = 732.6

a / b = 0.0747475

***Задание 1:*** ввести с клавиатуры массив, состоящий из n строк и m столбцов. Создать библиотеку, в которой будет содержатьсяфункция, которая производит вычисления согласно заданию, приведенному в таблице.

|  |  |  |  |
| --- | --- | --- | --- |
| № вар. | *n*\**m* | Тип элементов матрицы | Задача |
| 1 | 7\*4 | целый | Вычислить сумму индексов максимального элемента матрицы. |
| 2 | 8\*3 | вещественный | Вычислить среднее арифметическое элементов матрицы. |
| 3 | 4\*5 | целый | Сформировать одномерный массив как сумму элементов столбцов матрицы. |
| 4 | 3\*6 | вещественный | Вычислить сумму максимального и минимального элементов матрицы. |
| 5 | 6\*4 | вещественный | Сформировать одномерный массив как максимальные элементы строк матрицы. |
| 6 | 6\*6 | целый | Вычислить сумму максимального и минимального элементов главной диагонали матрицы. |
| 7 | 7\*5 | целый | Сформировать одномерный массив как сумму отрицательных элементов строк матрицы. |
| 8 | 5\*5 | вещественный | Вычислить среднее арифметическое элементов побочной диагонали матрицы. |
| 9 | 6\*6 | целый | Вычислить произведение элементов главной диагонали матрицы. |
| 10 | 5\*8 | целый | Получить вектор, как столбец матрицы с минимальной суммой элементов. |
| 11 | 7\*7 | вещественный | Поменять местами элементы главной и побочной диагоналей матрицы. |
| 12 | 5\*5 | вещественный | Сформировать одномерный массив как столбец матрицы, содержащий максимальный элемент на главной диагонали. |
| 13 | 7\*6 | целый | Сформировать одномерный массив как среднее арифметическое элементов столбцов матрицы. |
| 14 | 6\*5 | целый | Поменять местами максимальный и минимальный элементы матрицы. |
| 15 | 7\*7 | вещественный | Вычислить произведение максимальных элементов главной и побочной диагоналей матрицы. |
| 16 | 5\*6 | целый | Вычислить сумму элементов матрицы, находящихся в интервале [-2;3]. |
| 17 | 4\*8 | целый | Вычислить среднее геометрическое положительных элементов матрицы. |
| 18 | 8\*3 | целый | Вычислить произведение нечетных элементов матрицы. |
| 19 | 7\*7 | вещественный | Вычислить произведение положительных элементов, расположенных в секторе под главной диагональю матрицы. |
| 20 | 6\*4 | вещественный | Получить вектор, как столбец матрицы, содержащий максимальный элемент. |
| 21 | 7\*7 | вещественный | Вычислить сумму элементов матрицы, расположенных под главной диагональю. |
| 22 | 7\*5 | вещественный | Получить вектор, как строку матрицы с максимальной суммой элементов. |
| 23 | 4\*6 | вещественный | Сформировать одномерный массив как минимальные элементы столбцов матрицы. |
| 24 | 8\*3 | целый | Вычислить сумму элементов матрицы, кратных 3. |
| 25 | 7\*7 | вещественный | Поменять местами максимальный и минимальный элементы главной диагонали матрицы |
| 26 | 6\*4 | вещественный | Вычислить произведение элементов матрицы, находящихся в интервале [2;7]. |
| 27 | 7\*6 | целый | Получить вектор, как строку матрицы, содержащей максимальный элемент. |
| 28 | 5\*5 | вещественный | Сформировать одномерный массив как столбец матрицы, содержащий минимальный элемент на побочной диагонали. |
| 29 | 6\*6 | целый | Вычислить сумму модулей отрицательных элементов, расположенных в секторе над главной диагональю матрицы. |
| 30 | 6\*4 | вещественный | Сформировать одномерный массив как сумму отрицательных элементов строк матрицы. |

***Задание 2*:** написать программу согласно заданию.

