**Практическое занятие №8**

**«Функции»**

До сих пор в своих проектах мы использовали работу лишь с единственной функцией main. Использование дополнительных функций существенно увеличивает возможности программирования. Функция может реализовывать некоторые логические задачи, а затем многократно использоваться в различных программах. Функции могут создаваться квалифицированными программистами, по тщательно проработанным алгоритмам, включаться в библиотеки и затем использоваться там, где в этом возникнет необходимость. При создании сложных программ программа упрощается, если её удаётся разделить на отдельные модули и реализовать при помощи функций.

Язык C предоставляет обширные возможности для программирования функций.

Переменные, описанные внутри функции, являются локальными, они существуют только в момент использования функцию. Если нам нужно возвратить некоторое значение, то это делается в операторе **return**. Переменные, описанные вне функций, являются глобальными, они могут использоваться в любой функции, описанной после описания этих переменных.

Транслятор C традиционно является однопроходным, т.е. просматривает исходный текст программы один раз. Поэтому к моменту обращения к функции он должен знать прототип функции. Если описание функции следует после обращения к ней, то следует задать прототип функции. Тогда в точке обращения к функции, транслятор сформирует ссылку на функцию и продолжит работу. Окончательное формирование исполнительного кода программы произойдёт после, на этапе редактирования. Сами функции могут быть взяты из входного потока или из библиотек. Если мы уберём описание нашей функции, то сообщение об ошибке поступит уже от редактора.

Что происходит при обращении к функции

Если внутри программы произошло обращение к функции, то транслятор сформирует соответствующую ссылку. В момент обращения уже в процессе выполнения, список входных параметров будет занесен в специальный стек, после чего вершина стека будет перенесена за этот список, после чего управление будет физически передано вызываемой программе. Если вызываемая программа, в свою очередь, обращается к другим функциям, то вызов к ним будет осуществлен аналогично. По завершении функции вершина стека возвращается в то состояние, которое было до вызова функции. Эта методика обеспечивает то обстоятельство, что мы не можем изменить входные параметры изнутри вызываемой функции.

Модификатор inline

Если отметить функцию при помощи модификатора inline, то тело функции будет включаться в тело вызывающей программы в месте обращения; таким образом, загрузки и освобождение стека производиться не будет.

**Пример 1.** Создать проект. Объявить и вызвать функцию, возвращающую первое из двух значений, представленных параметрами, кратное 5. Если оба значения не кратны 5, то функция возвращает -1 **(lab8\_1.cpp).**

**Пример 2.** Создать функцию вычисления суммы и произведения двух чисел с использованием глобальных переменных **(lab8\_2.cpp).**

**Пример 3.** Реализовать функцию void, возвращающую через параметры-указатели в качестве результата сумму и произведение двух чисел, представленных параметрами-значениями **(lab8\_3.cpp).**

**Пример 4.** Создать программу, содержащую функцию выборки из одномерного массива всех отрицательных элементов и сохранении их в другом одномерном массиве **(lab8\_4.cpp).**

**Задание 1.** Создать проект. Реализовать функцию, которая позволяет получать числа типа double, при помощи генератора случайных чисел. Функция имеет прототип: double randdouble(). Для отладки вывести несколько полученных чисел на консоль.

**Задание 2.** Создать проект. Реализовать функцию выбора минимального числа из двух тремя способами: с помощью возвращаемого значения функции, с помощью глобальных переменных, с помощью параметров-указателей.

1. **Перегружаемые функции**

В рамках одной программы можно объявить несколько функций с одинаковыми именами, но различающихся по типу или числу параметров. Компилятор после анализа фактических параметров в вызове подпрограммы выберет подходящий вариант подпрограммы из числа одноименных подпрограмм. Например, при делении целых чисел их надо делить нацело, а при делении действительных чисел требуется получать результат действительного типа **(lab8\_5.cpp).**

В этом случае следует объявить две одноименных функции деления, у которых будет различный тип аргументов, а также будет различаться тип результата.

1. **Параметры со значениями по умолчанию**

Для параметров в объявлении подпрограмм можно задавать параметры по умолчанию. Значение по умолчанию – это значение параметра, которое используется при вызове подпрограммы, если в нее не передано фактическое значение. Использование параметров по умолчанию равносильно разрешению указывать в вызове подпрограммы не все необходимые параметры.

Значения по умолчанию задаются добавлением после объявления типа формального параметра знака равенства, после которого записывается константное выражение. Рассмотрим функцию, которая рассчитывает силу тяжести бруска в форме прямоугольного параллелепипеда в зависимости от его размеров, материала (плотности) и ускорения свободного падения **(lab8\_6.cpp).**

1. **Передача имен функций в качестве параметров**

При решении некоторых задач появляется необходимость передачи в функцию в качестве параметра имени другой функции. Например, требуется вычислить значение следующего выражения **(lab8\_7.cpp**):



Для получения результата надо трижды вычислять сумму, для этого целесообразно использовать функцию. Однако следует заметить, что функция должна вычислять не просто сумму элементов массива, а сумму значений некоторой функции от элементов массива. Поэтому наряду с массивом в функцию надо передавать имя функции, вычисляющей очередное слагаемое.

Функция может быть вызвана с помощью указателя на нее. Для этого надо объявить указатель соответствующего типа и присвоить ему адрес необходимой функции с помощью операции взятия адреса. Например:

*//объявляем pf как указатель на функцию типа float любоеИмя(float)*

*typedef float (\*pf)(float);*

*//определяем функцию fun такого типа*

*float fun(float) {/\* тело функции \*./}*

*pf=&fun; //получаем адрес функции fun*

*c=pf(a);//вызываем функцию fun по указателю pf на неё*

Удобно использовать переименование типов (typedef) при описании указателей на функцию. Указатели на функцию передаются в подпрограмму так же, как и параметры других типов. Тип указателя и тип функции, передаваемой посредством этого указателя, должны совпадать полностью.

1. **Использование модульного подхода (lab8\_8.cpp)**

Модульный подход подразумевает разбиение всей программы на несколько физических файлов – в данном примере на 3 файла: 1 заголовочный файл (формат .h) с созданием новых типов данных (слово typedef) и с заголовками используемых функций; 2 файла исходного кода (формат .cpp) – 1 файл с самими функциями, 2 файл – с функцией main().

**Описание примера:** Удалить из матрицы B(M,N) действительных чисел строки, содержащие максимальный и минимальный элементы всей матрицы. Если минимальный и максимальный элементы находятся в одной строке, то никакие строки не удалять. Строки преобразованной матрицы упорядочить методом пузырька по возрастанию сумм цифр целой части последних элементов строк матрицы.

Приступая к составлению модуля, необходимо определить основные задачи, решение которых позволит решить общую задачу.

1. Ввод исходных данных – количества строк и столбцов матрицы, а также самих элементов матрицы (подпрограмма ввода исходных данных).

2. Вывод исходных данных для последующего контроля правильности решения задачи (подпрограмма вывода матрицы).

3. Определение номера строки матрицы, содержащей максимальный элемент матрицы, т.е. поиск максимального элемента всей матрицы и номера строки, в которой он расположен (подпрограмма поиска максимального элемента).

4. Определение номера строки матрицы, содержащей минимальный элемент матрицы, т.е. поиск минимального элемента всей матрицы и номера строки, в которой он расположен (подпрограмма поиска минимального элемента).

5. Удаление из матрицы строки с заданным номером.

6. Определение цифр целого числа.

7. Вычисление суммы элементов одномерного массива.

8. Сортировка строк матрицы.

9. Вывод полученных результатов.

**Задание 3.** Определить в каждой строке матрицы A(m,n), m<=10, n<=12, первое по порядку простое число и занести его в одномерный массив. Если в строке нет простых чисел, то для этой строки в массив занести 0. Написать программу, используя модульный подход.

**Задание 4.** Составить функцию, которая в каждой строке матрицы заменяет каждый положительный элемент минимальным отрицательным элементом из стоящих между предыдущим и текущим положительными элементами. Если положительный элемент стоит на первом месте в строке, то его не изменять. Если отрицательных элементов нет между положительными, то замену не производить. Использовать составленную функцию для матрицы W(m,n),m<=12,n<=15. Написать программу, используя модульный подход.