**Глава 10. Функции в С++. Параметры функций. Ссылки и указатели. Модификаторы функций и переменных**

**10.1 Функции в C++**

Функции – это именованные блоки кода, предназначенные для выполнения определенных операций. Функции служат ключевым инструментом для структурирования программного кода. Функции работают таким образом, что в них помещаются входные данные, обрабатываются и на их основе появляются выходные данные, которые возвращаются из функции.

Объявление функции, которое также можно назвать прототипом, заголовком или сигнатурой задает ее имя, тип возвращаемого значения и список передаваемых параметров. Определение функции содержит, кроме объявления, тело функции, представляющее собой последовательность операторов и описаний в фигурных скобках. На рисунке 10.1 представлен синтаксис определения функции.

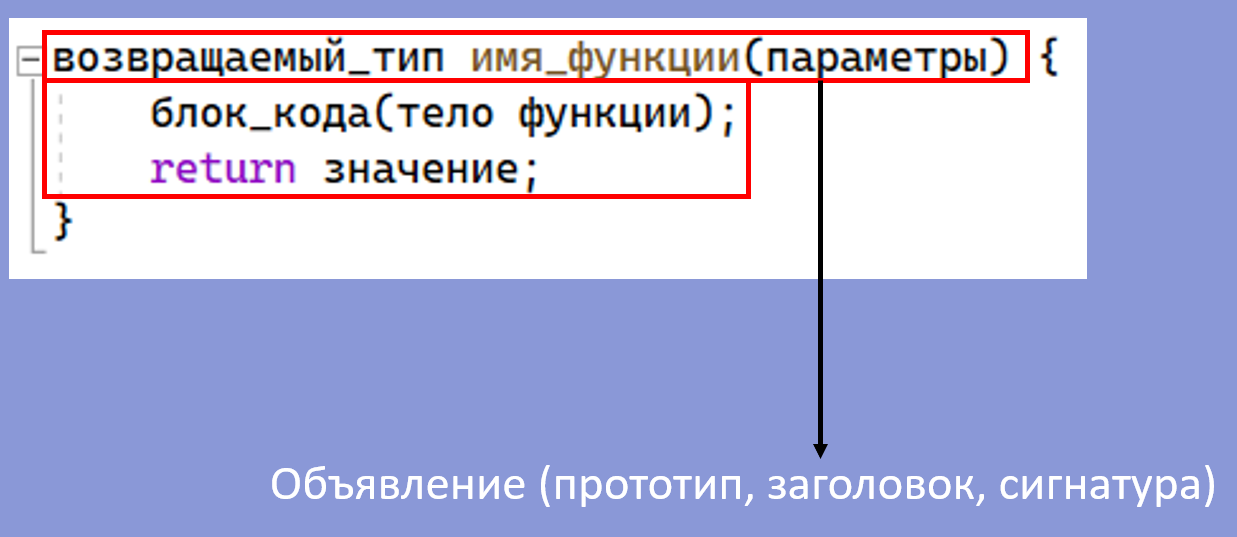


Рисунок 10.1 – Структура объявления и определения функции

**10.2 Функция main()**

Необходимая функция, которая пишется первой – функция “main()”. Данная функция обозначает точку начала выполнения программы. Возвращаемое значение должно быть целого типа. Стандарт предусматривает два формата функции: с параметрами и без параметров (рисунок 10.2). Имена параметров в программе могут быть любыми, но принято использовать argc и argv. Argc принимает количество аргументов командной строки, а argv – массив строк, содержащий аргументы командной строки.

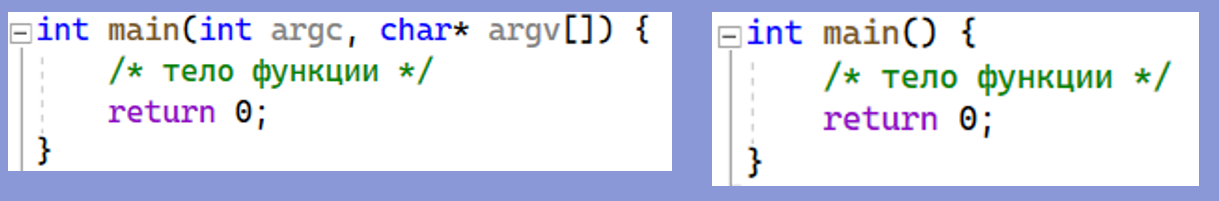


Рисунок 10.2 – Виды функции main()

**10.3 Возвращаемый тип функции**

Возвращаемый тип функции определяет тип данных, который функция возвращает после своего выполнения. Данный тип бывает любым встроенным или пользовательским типом данных. При завершении работы функция возвращает значение определенного типа данных обратно в вызывающую ее часть программы. Механизм возврата из функции реализуется с помощью оператора return. В примере, представленном на рисунке 10.3, функция считает сумму двух целых чисел, соответственно её возвращаемое значение будет типа int.

Нельзя возвращать из функции указатель на локальную переменную, поскольку память, выделенная локальными переменными при входе в функцию, освобождается после возврата из нее.

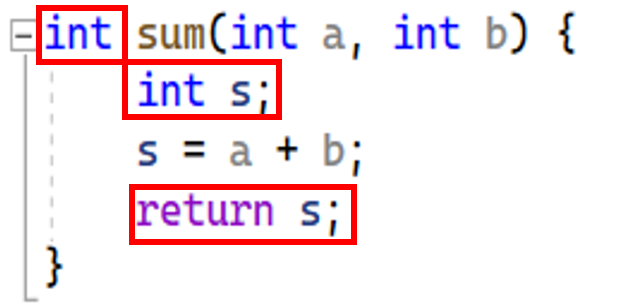


Рисунок 10.3 – Функция sum возвращает значение целого типа суммы параметров целого типа а и b

Функция типа void используется в объявлении для обозначения того, что функция не возвращает значение. Такие функции выполняют определенные действия или операции, но не предоставляют результат в виде значения. В таких функциях оператор return можно опускать, если возврат происходит перед закрывающей фигурной скобкой. На рисунке 10.4 показано, что функция считает сумму двух чисел, но при этом не возвращает значение, а сразу выводит результат на консоль.

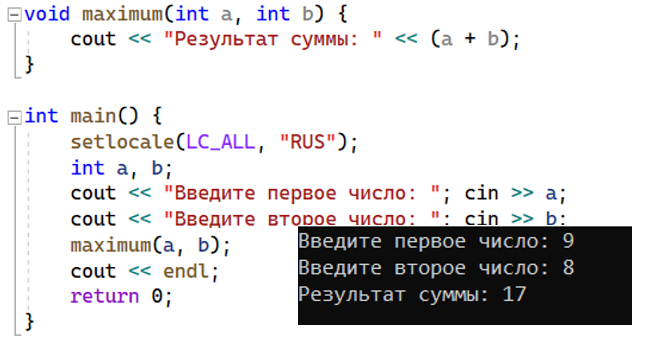


Рисунок 10.4 – Пример функции типа void

**10.4 Параметры функции**

Параметры функции – это переменные, которые передаются в функцию при ее вызове. Параметры функции передают данные в функцию, с которыми функция работает. Параметры определяются в заголовке функции и могут быть использованы внутри тела функции.

Параметры, перечисленные в заголовке, называются формальными параметрами, или просто параметрами, а записанные в операторе вызова функции — фактическими параметрами, или аргументами [17].

Имена формальных и фактических параметров могут не совпадать, но при этом обязательно должно совпадать их количество и типы. На представленном рисунке 10.5 фактическими параметрами являются first и second, при передаче в функцию данные параметры подставляются вместо параметров a и b. Значение first встает на место a, значение second – на место b.

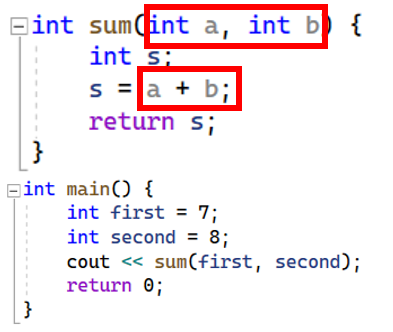


Рисунок 10.5 – Пример использования параметров функции

**10.5 Передача параметров в функцию**

Существует два способа передачи параметров в функцию: по значению и по адресу (передача по адресу осуществляется с помощью ссылок или указателей).

При передаче *по значению* операторы функции работают с *копиями аргументов*. Доступа к исходным значениям параметров у функции нет, а, следовательно, нет и возможности их изменить. Поэтому после завершения функции значения аргументов в памяти компьютера остаются исходными. Изменить аргументы, можно только присвоив им значение, которое вернет функция (рисунок 10.6).

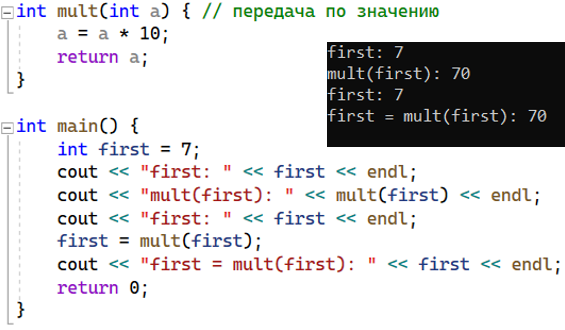


Рисунок 10.6 – Пример передачи параметров в функцию по значению

При передаче *по адресу* функция осуществляет доступ к *ячейкам памяти* по этим адресам и может изменить исходные значения аргументов. При передаче по ссылке в функцию передается адрес указанного при вызове параметра, а внутри функции все обращения к параметру неявно получают доступ к значению параметра или разыменовываются. Использование ссылок вместо указателей улучшает читаемость программы, избавляя от необходимости применять операции получения адреса и разыменования. Примеры передачи параметров по адресу представлены на рисунках 10.7 и 10.8.

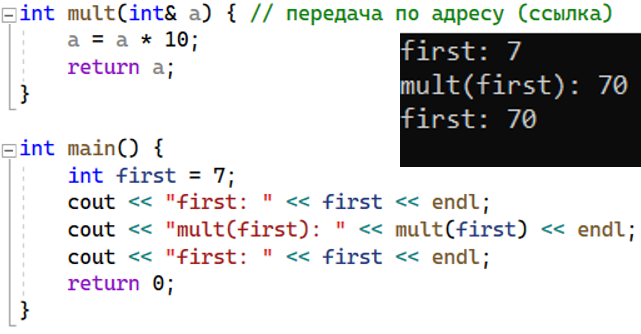


Рисунок 10.7 – Пример передачи параметров в функцию по адресу (ссылка)

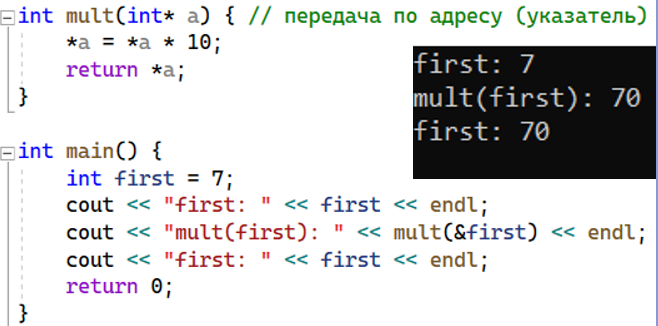


Рисунок 10.8 – Пример передачи параметров в функцию по адресу (указатель)

При использовании в качестве параметра массива, в функцию передается указатель на его первый элемент [33]. При этом информация о количестве элементов массива теряется, и следует передавать его размерность через отдельный параметр (рисунок 10.9).

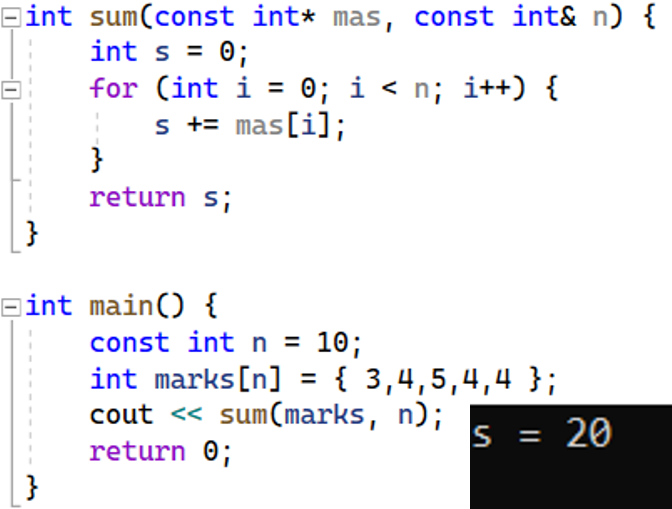


Рисунок 10.9 – Пример передачи массива в функцию

**10.6 Модификаторы функций**

Модификаторы функций – это специальные ключевые слова, которые изменяют поведение функций и их взаимодействие с данными и вызывающим кодом. Модификаторы позволяют дополнительно настраивать функции в соответствии с требованиями программы.

Модификатор «inline» предоставляет рекомендацию компилятору встроить код функции непосредственно в место вызова, вместо обычного вызова функции [32]. Это может улучшить производительность, уменьшить накладные расходы на вызов функции и сделать код более эффективным. Модификатор «inline» часто используется для небольших и простых функций (рисунок 10.10).

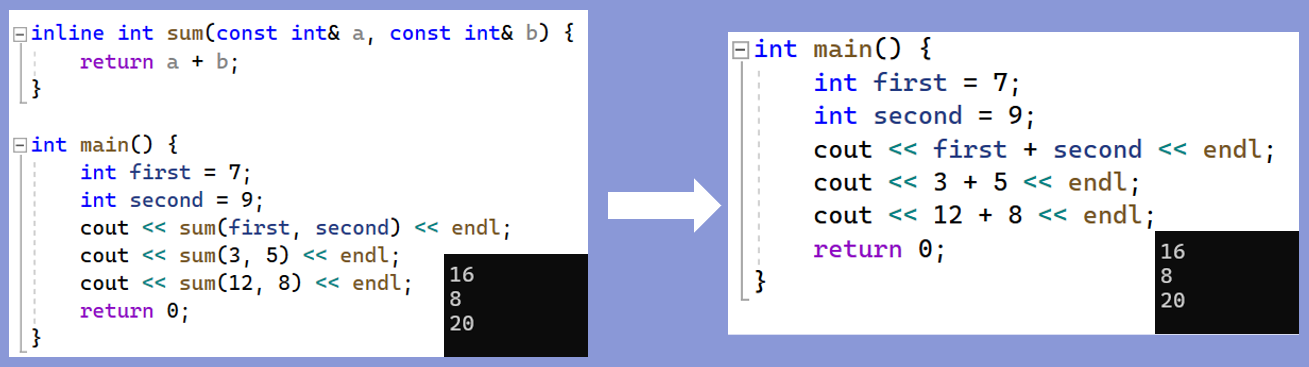


Рисунок 10.10 – Пример функции с модификатором «inline»

Модификатор «static» ограничивает область видимости функции только текущим файлом, в котором она определена. Такая функция называется "файловой функцией" или "локальной функцией". Файловая функция с модификатором «static» доступна только в пределах текущего файла и не может быть использована из других файлов (рисунок 10.11). Если одна и та же функция с модификатором «static» определена в разных файлах, каждый из этих файлов будет иметь свою собственную копию функции, и они не будут взаимодействовать между собой.

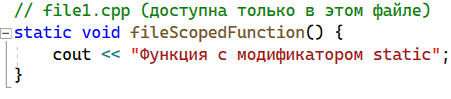


Рисунок 10.11 – Пример функции с модификатором «static»

Модификатор «extern» позволяет объявлять функции в одном файле и использовать их в других файлах (рисунок 10.12). Это удобно для разделения кода и создания более организованных и масштабируемых проектов. Однако следует учитывать, что использование «extern» может потребовать внимания к правильному порядку компиляции и связывания файлов. Также модификатор «extern» может использоваться, когда прототип функции в одном файле, а объявление в другом.

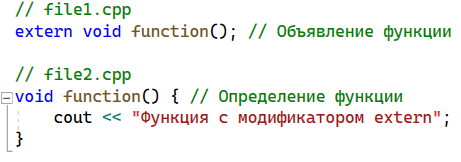


Рисунок 10.12 – Пример функции с модификатором «extern»

**10.7 Модификаторы переменных. Модификатор «static»**

Также помимо функций некоторые из модификаторов можно использовать для переменных. Все величины, описанные внутри функции, а также ее параметры, являются локальными. Областью их действия является функция. Значения локальных переменных между вызовами одной и той же функции не сохраняются. Если этого требуется избежать, при объявлении локальных переменных используется модификатор «static». На рисунке 10.13 статическая переменная n размещается в сегменте данных и инициализируется один раз при первом выполнении оператора, содержащего ее определение. Автоматическая переменная или переменная с модификатором «auto» m инициализируется при каждом входе в функцию. Автоматическая переменная р инициализируется при каждом входе в блок цикла. Модификатор «auto» устанавливается по умолчанию при объявлении переменной и обозначает, что переменная будет иметь тип, определяемый компилятором на основе инициализатора, что делает код более чистым и гибким.

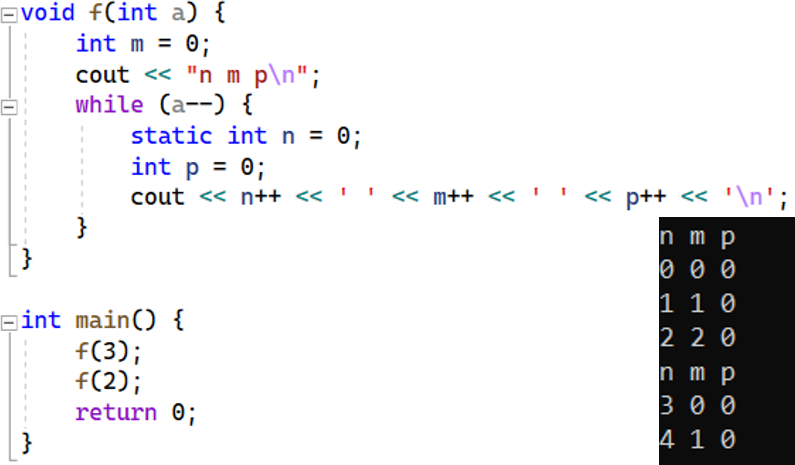


Рисунок 10.13 – Пример работы переменной с модификатором «static»