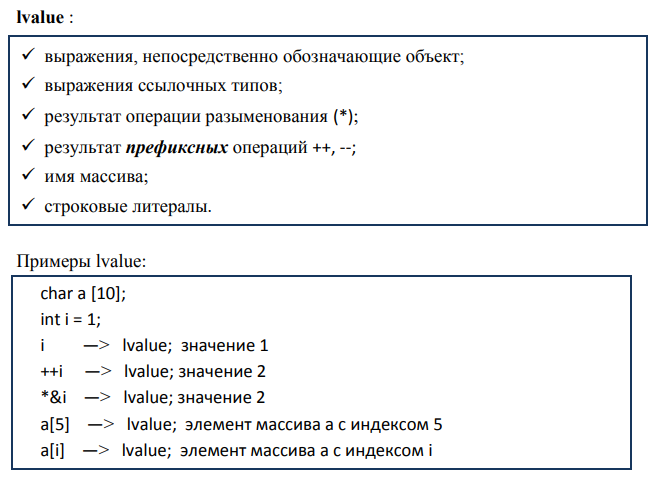
1. Структура языка программирования: определение выражения, его состав, порядок вычисления выражения. Символ окончания последовательности. Примеры (С++).

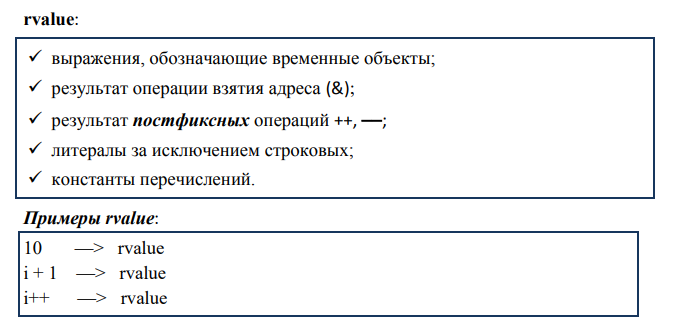
Выражение – объединение литералов, имен (переменных, функций и пр.), операторов и специальных символов, служащих для вычисления выражения или достижения побочных эффектов (например: при применении в выражении функций). Выражения состоят из операндов, знаков операций и скобок. Порядок вычисления выражения с операторами одинакового приоритета не определен. Символ окончания последовательности (в языке программирования C++ – это точка с запятой) определяет точку последовательности (sequence point), в которой завершились все вычисления и побочные эффекты. Точка последовательности – момент времени, когда побочные эффекты вычисленных выражений уже случились, а побочные эффекты следующих в последовательности выражений еще не начались.

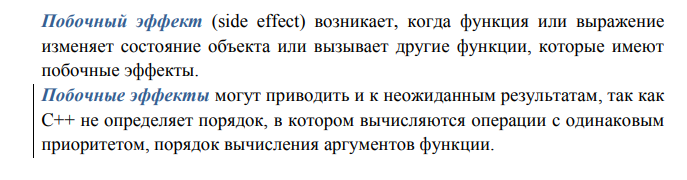
1. Структура языка программирования: выражения (lvalue, rvalue, побочные эффекты, точка последовательности, унарные, бинарные и тернарные выражения), константные выражения, укороченное вычисление. Примеры (С++).

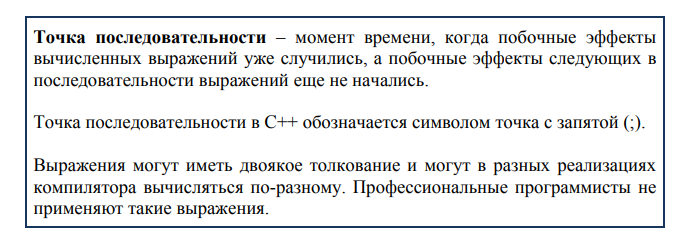
lvalue (именующее выражение) – это ссылка на значение; могут использоваться в левой и правой части оператора присваивания. Имя переменной, ссылка на элемент массива по индексу, вызов функции возвращающей указатель, всегда связаны с областью памяти, адрес которой известен

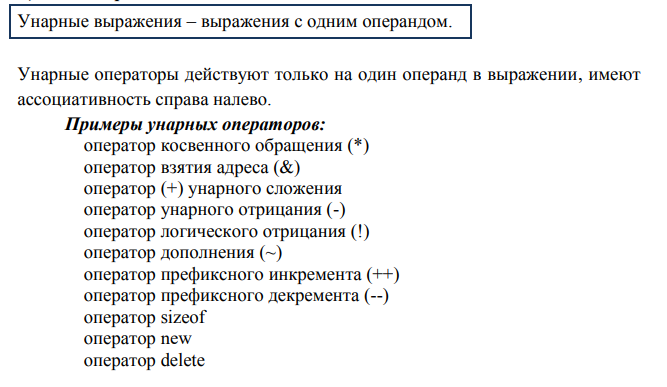


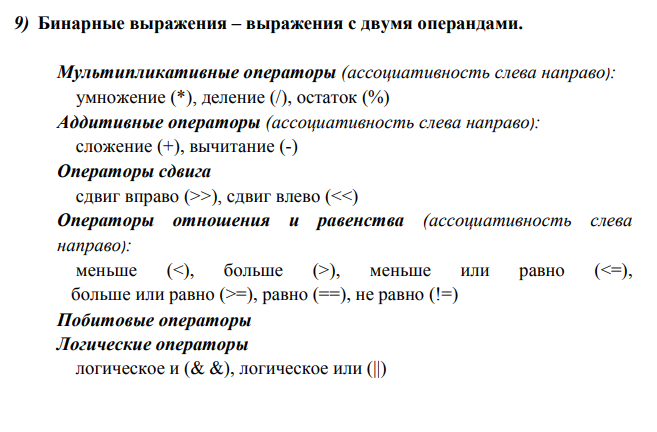
rvalue (значащее выражение) – может использоваться только в правой части оператора присваивания (не связано с адресом, связано только со значением; это могут быть литералы, вызов функции, возвращающей значение).

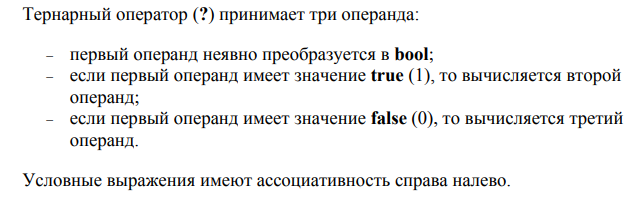












1. Структура языка программирования: инструкции языка программирования, объявление, определение, инициализация, простые и составные инструкции, инструкции выбора, инструкции циклов, инструкции переходов, примеры (С++).

**Инструкция (оператор)** — наименьшая автономная часть языка программирования; команда или набор команд. Программа обычно представляет собой последовательность инструкций.

**Объявление** включает в себя указание идентификатора, типа, а также других аспектов элементов языка, например, переменных и функций. Объявление используется, чтобы уведомить компилятор о существовании элемента. (Определение – реализация, при этом компилятор выделяет место. Переменная может быть объявлена несколько раз, а определена только один раз. Объявление предоставляет основные свойства символа: его тип и его название. Определение предоставляет все детали этого символа — если это функция, что она делает; если это класс, какие у него поля и методы; если это переменная, где эта переменная находится. Инициализация – придание начального значения переменной при её объявлении)

**Простые инструкции** завершаются точкой с запятой. **Составные инструкции**, или блоки ({ }), состоят из одной или более инструкций (каждая из которых сама может быть составной), заключенных в фигурные скобки.

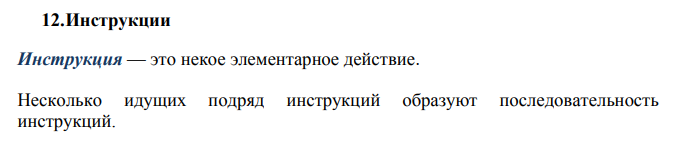
**Инструкции выбора** *if*и *switch*, позволяют выполнять те или иные участки кода в зависимости от выполнения условий.

**Инструкции циклов** используются для многократного повторения фрагментов кода. Цикл *for* мы можем использовать, если знаем точное количество действий (итераций). Когда мы не знаем, сколько итераций должен произвести цикл, нам понадобится цикл *while* или *do…while*. Цикл *do…while* почти ничем не отличается от цикла *while*, за исключением того, что тело цикла гарантированно выполняется хотя бы один раз.

**Инструкции переходов** выполняют немедленную локальную передачу контроля. Оператор *break* завершает выполнение ближайшего внешнего цикла или условного оператора, в котором он находится. Управление передается оператору, который расположен после оператора, при его наличии. Инструкцию *continue* можно располагать только внутри цикла. Она вызывает переход к следующей итерации самого внутреннего содержащего ее цикла. С помощью *return* функция возвращает управление в программу, откуда была вызвана. Если за *return* следует выражение, то его значение возвращается вызвавшей эту функцию программе. Оператор *goto* осуществляет безусловную передачу управления оператору, метка которого задана идентификатором. Метка оператора имеет значение только для оператора *goto*; в остальных случаях метки операторов игнорируются. Повторное объявление меток невозможно.

**Инструкции обработки исключений** используют три ключевых слова: *try*, *catch* и *throw*. Те инструкции программы, где ожидается возможность появления исключительных ситуаций, содержатся в бло­ке *try*. Если в блоке *try* возникает исключение, т. е. ошибка, то генерируется исключение throw. Исклю­чение перехватывается, используя *catch*, и обрабатывается.

1. Структура языка программирования: инструкции языка программирования, инструкции обработки исключений, примеры (С++).



**Инструкции выбора** *if*и *switch*, позволяют выполнять те или иные участки кода в зависимости от выполнения условий.

**Инструкции циклов** используются для многократного повторения фрагментов кода. Цикл *for* мы можем использовать, если знаем точное количество действий (итераций). Когда мы не знаем, сколько итераций должен произвести цикл, нам понадобится цикл *while* или *do…while*. Цикл *do…while* почти ничем не отличается от цикла *while*, за исключением того, что тело цикла гарантированно выполняется хотя бы один раз.

**Инструкции переходов** выполняют немедленную локальную передачу контроля. Оператор *break* завершает выполнение ближайшего внешнего цикла или условного оператора, в котором он находится. Управление передается оператору, который расположен после оператора, при его наличии. Инструкцию *continue* можно располагать только внутри цикла. Она вызывает переход к следующей итерации самого внутреннего содержащего ее цикла. С помощью *return* функция возвращает управление в программу, откуда была вызвана. Если за *return* следует выражение, то его значение возвращается вызвавшей эту функцию программе. Оператор *goto* осуществляет безусловную передачу управления оператору, метка которого задана идентификатором. Метка оператора имеет значение только для оператора *goto*; в остальных случаях метки операторов игнорируются. Повторное объявление меток невозможно.

**Инструкции обработки исключений** используют три ключевых слова: *try*, *catch* и *throw*. Те инструкции программы, где ожидается возможность появления исключительных ситуаций, содержатся в бло­ке *try*. Если в блоке *try* возникает исключение, т. е. ошибка, то генерируется исключение throw. Исклю­чение перехватывается, используя *catch*, и обрабатывается.

1. Структура языка программирования: программные конструкции (блоки, функции, процедуры и пр.). Функции: передача параметров по значению и по ссылке, передача переменного числа параметров в функции C++. Примеры.

**Программные конструкции:**

**Блоки**: логически сгруппированный набор идущих подряд инструкций в [исходном коде](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) программы, является основой парадигмы [структурного программирования](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)

**Пример С++:**

{

int a = 1;

int b = 2;

int c = a + b;

return c;

}

**Функции:** фрагмент программного кода, к которому можно обратиться из другого места программы

**Пример С++:**

void name(char\* text)

{

std::cout<<text<< std::endl;

}

**Процедуры:** функции не возвращающие значения, return не обязателен, но желателен для явного указания завершения функции

**Пример С++:**

 void ValidMessage()

 {

         cout << "Пароль введен верно" << endl;

 }

**Передача параметров:** передача параметров в функцию происходит через стек. Код, вызывающий функцию, знает, сколько параметров ей передать и каковы значения этих параметров

**Передача параметров**

**по значению:** они при выходе из функции не изменятся

**Пример С++:**

int func(int k){

k\*=2;

return k;

}

void main(){

int z=1, y=3, k;

k=func(z)+func(y);

cout<<z<<" "<<y;

}

Z = 1, y = 3

**по ссылке:** при выходе из функции из значения могут измениться

int func(int &k){

k\*=2;

return k;

}

void main(){

int z=1, y=3, k;

k=func(z)+func(y);

cout<<z<<" "<<y;

}

Z = 2, y = 6

**Передача переменного числа параметров в функции С++:**

По умолчанию параметры передаются в функцию через стек. Поэтому, технически, нет ограничения на количество передаваемых параметров. Функции с переменным числом параметров объявляются как обычные функции, но вместо недостающих аргументов ставится многоточие. Количество параметров и их типы известны только при вызове функции.

Передать функции параметры можно двумя способами:

* явно передать обязательный аргумент, задающий число параметров;
* добавить в конец списка параметр с уникальным значением, по которому будет определяться конец списка параметров.

Общий принцип работы следующий: в функции для доступа к списку параметров устанавливается указатель, значением которого будет адрес явного параметра в списке, далее изменяется значение этого указателя, чтобы переместиться на следующий параметр.

1. Структура языка программирования: программные конструкции (блоки, функции, процедуры и пр.). Передача параметров в функцию. Передача параметров по значению и по ссылке, передача значений параметров по умолчанию в функции C++. Примеры.

**Программные конструкции:**

**Блоки**: логически сгруппированный набор идущих подряд инструкций в [исходном коде](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) программы, является основой парадигмы [структурного программирования](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)

**Пример С++:**

{

int a = 1;

int b = 2;

int c = a + b;

return c;

}

**Функции:** фрагмент программного кода, к которому можно обратиться из другого места программы

**Пример С++:**

void name(char\* text)

{

std::cout<<text<< std::endl;

}

**Процедуры:** функции не возвращающие значения, return не обязателен, но желателен для явного указания завершения функции

**Пример С++:**

 void ValidMessage()

 {

         cout << "Пароль введен верно" << endl;

 }

**Передача параметров:** передача параметров в функцию происходит через стек. Код, вызывающий функцию, знает, сколько параметров ей передать и каковы значения этих параметров

**Передача параметров**

**по значению:** они при выходе из функции не изменятся

**Пример С++:**

int func(int k){

k\*=2;

return k;

}

void main(){

int z=1, y=3, k;

k=func(z)+func(y);

cout<<z<<" "<<y;

}

Z = 1, y = 3

**по ссылке:** при выходе из функции из значения могут измениться

int func(int &k){

k\*=2;

return k;

}

void main(){

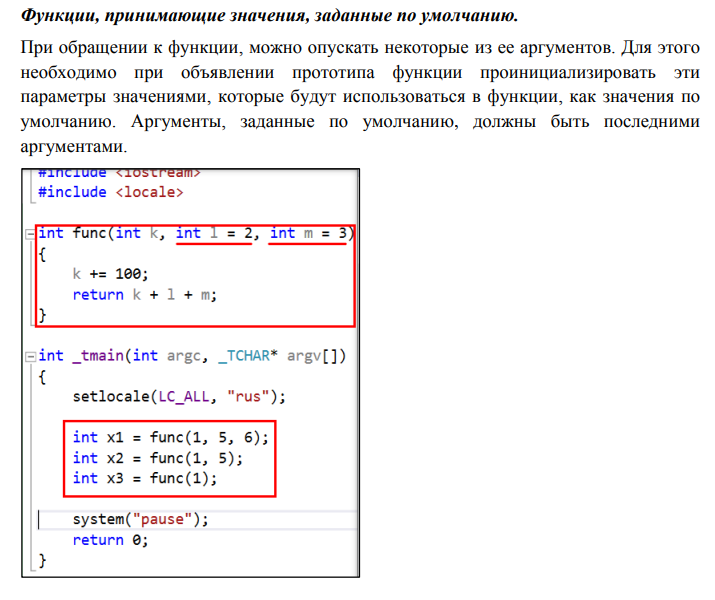
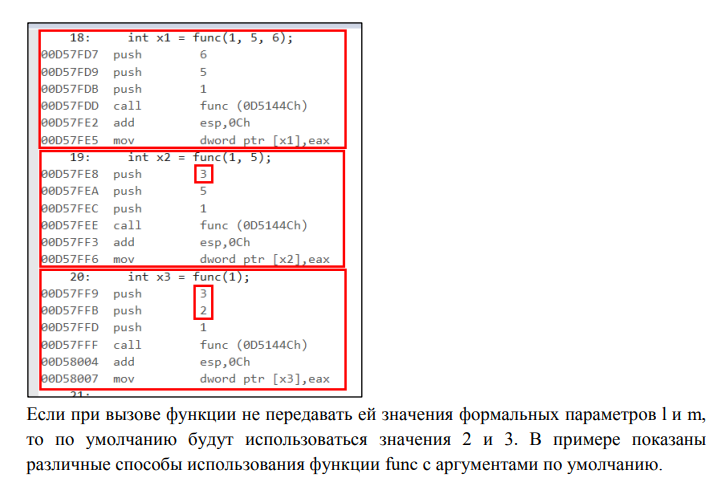
int z=1, y=3, k;

k=func(z)+func(y);

cout<<z<<" "<<y;

}

Z = 2, y = 6

1. Структура языка программирования: программные конструкции (блоки, функции, процедуры и пр.). Передача параметров в функцию. Реализация программных конструкций: лямбда-функции в С++. Примеры.

**Блоки**: логически сгруппированный набор идущих подряд инструкций в [исходном коде](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) программы, является основой парадигмы [структурного программирования](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)

**Пример С++:**

{

int a = 1;

int b = 2;

int c = a + b;

return c;

}

**Функции:** фрагмент программного кода, к которому можно обратиться из другого места программы

**Пример С++:**

void name(char\* text)

{

std::cout<<text<< std::endl;

}

**Процедуры:** функции не возвращающие значения, return не обязателен, но желателен для явного указания завершения функции

**Пример С++:**

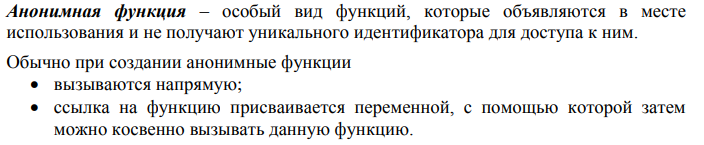
 void ValidMessage()

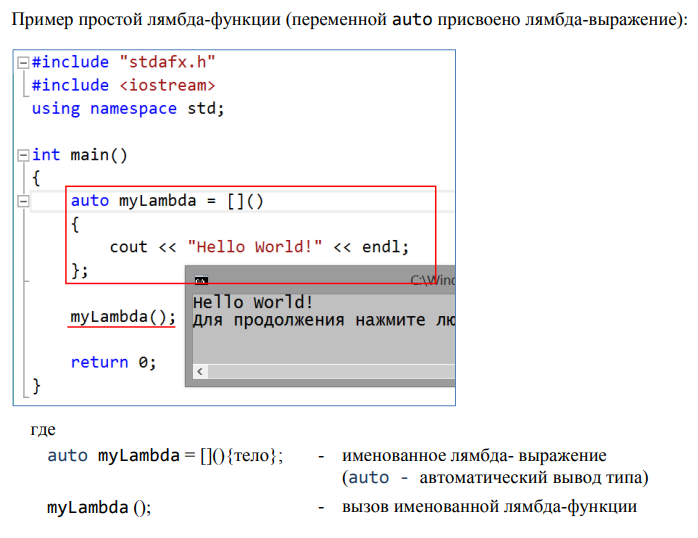
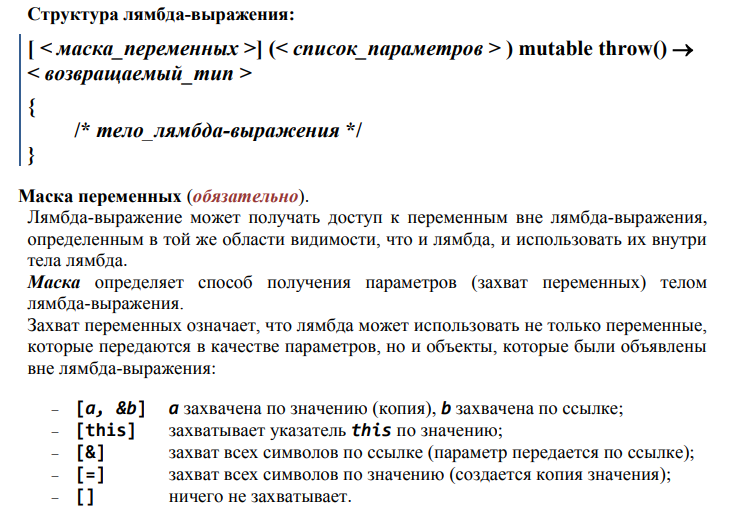
 {

         cout << "Пароль введен верно" << endl;

 }

**Передача параметров:** передача параметров в функцию происходит через стек. Код, вызывающий функцию, знает, сколько параметров ей передать и каковы значения этих параметров



1. Структура языка программирования: программные конструкции (блоки, функции, процедуры и пр.). Передача параметров в функцию. Реализация программных конструкций в C++. Перегружаемые функции в C++. Inline-функции в C++. Шаблоны функций С++.

**Блоки**: логически сгруппированный набор идущих подряд инструкций в [исходном коде](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) программы, является основой парадигмы [структурного программирования](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)

**Пример С++:**

{

int a = 1;

int b = 2;

int c = a + b;

return c;

}

**Функции:** фрагмент программного кода, к которому можно обратиться из другого места программы

**Пример С++:**

void name(char\* text)

{

std::cout<<text<< std::endl;

}

**Процедуры:** функции не возвращающие значения, return не обязателен, но желателен для явного указания завершения функции

**Пример С++:**

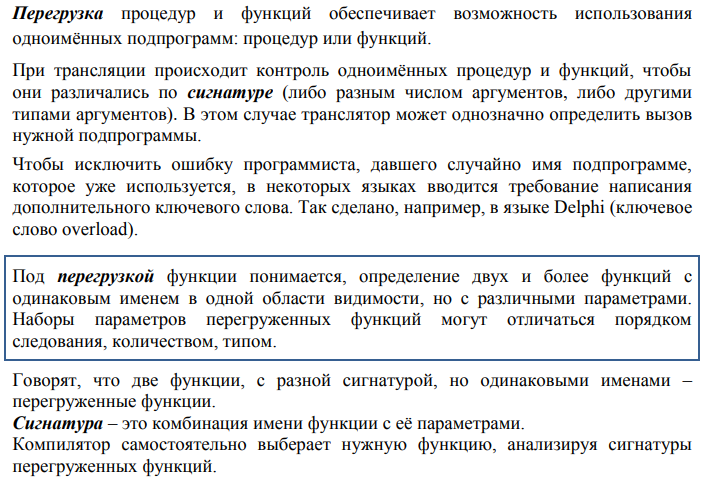
 void ValidMessage()

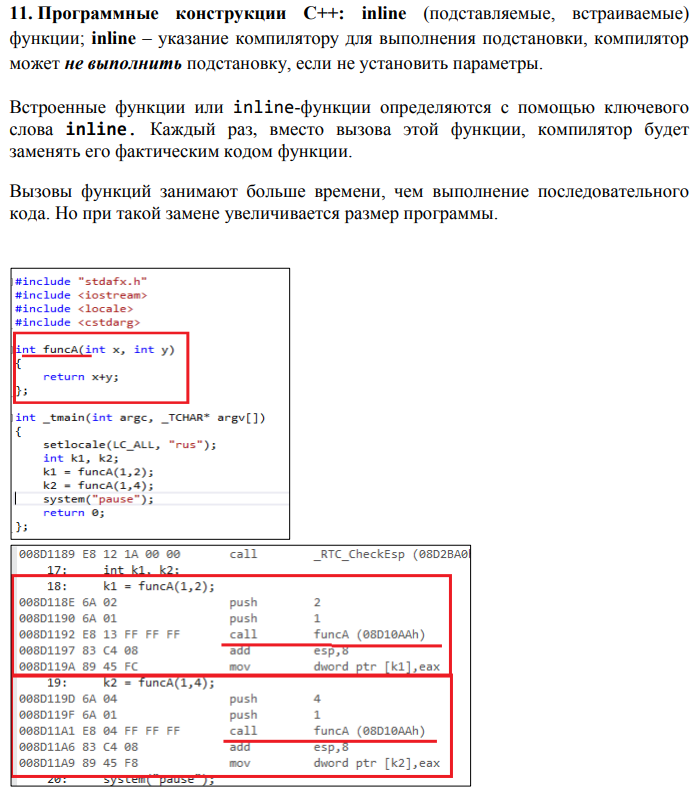
 {

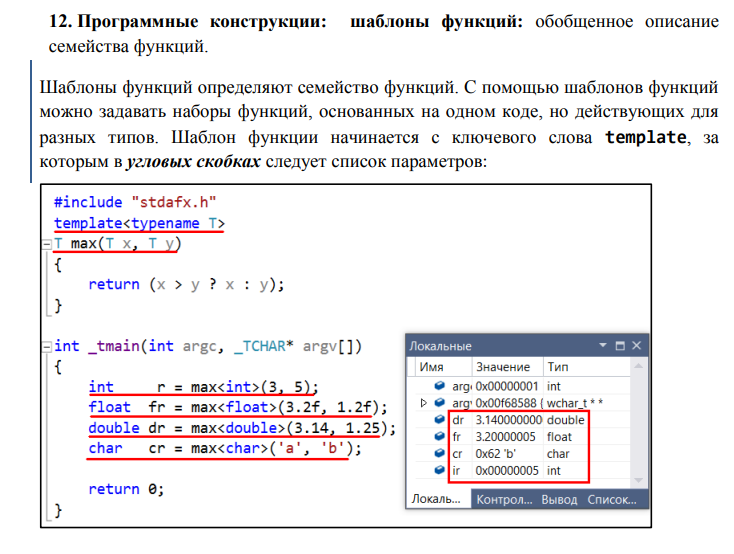
         cout << "Пароль введен верно" << endl;

 }

**Передача параметров:** передача параметров в функцию происходит через стек. Код, вызывающий функцию, знает, сколько параметров ей передать и каковы значения этих параметров







1. Структура языка программирования: программные конструкции (блоки, функции, процедуры и пр.). Передача параметров в функцию. Реализация программных конструкций в C++: соглашения о вызовах (\_cdecl).

**Блоки**: логически сгруппированный набор идущих подряд инструкций в [исходном коде](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) программы, является основой парадигмы [структурного программирования](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)

**Пример С++:**

{

int a = 1;

int b = 2;

int c = a + b;

return c;

}

**Функции:** фрагмент программного кода, к которому можно обратиться из другого места программы

**Пример С++:**

void name(char\* text)

{

std::cout<<text<< std::endl;

}

**Процедуры:** функции не возвращающие значения, return не обязателен, но желателен для явного указания завершения функции

**Пример С++:**

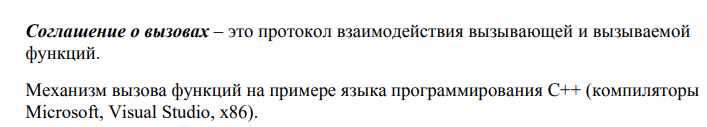
 void ValidMessage()

 {

         cout << "Пароль введен верно" << endl;

 }

**Передача параметров:** передача параметров в функцию происходит через стек. Код, вызывающий функцию, знает, сколько параметров ей передать и каковы значения этих параметров



**\_cdecl:** Параметры функций помещаются в стек, порядок параметров "справа налево". Параметры, размер которых меньше 4-х байт, расширяются до 4-х байт. Адрес возврата кладется в стек поверх параметров.

1. Структура языка программирования: программные конструкции (блоки, функции, процедуры и пр.). Передача параметров в функцию. Реализация программных конструкций в C++: соглашения о вызовах (\_stdcall).

**Блоки**: логически сгруппированный набор идущих подряд инструкций в [исходном коде](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) программы, является основой парадигмы [структурного программирования](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)

**Пример С++:**

{

int a = 1;

int b = 2;

int c = a + b;

return c;

}

**Функции:** фрагмент программного кода, к которому можно обратиться из другого места программы

**Пример С++:**

void name(char\* text)

{

std::cout<<text<< std::endl;

}

**Процедуры:** функции не возвращающие значения, return не обязателен, но желателен для явного указания завершения функции

**Пример С++:**

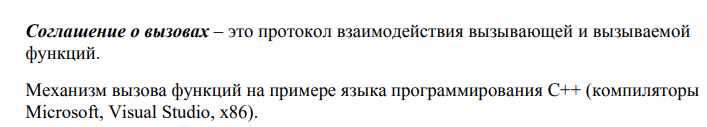
 void ValidMessage()

 {

         cout << "Пароль введен верно" << endl;

 }

**Передача параметров:** передача параметров в функцию происходит через стек. Код, вызывающий функцию, знает, сколько параметров ей передать и каковы значения этих параметров



**\_stdcall:** Параметры помещаются в стек, порядок параметров "справа налево". Адрес возврата кладется в стек поверх параметров.

Стек освобождает вызываемый код. В последней инструкции вызываемого кода RET указывается суммарный размер в байтах всех параметров процедуры. Команда RET после извлечения адреса возврата прибавляет к регистру ESP указанное значение. Целостность стека восстанавливается вызываемым кодом.

Возвращаемый параметр передается через регистр EAX.