**1. Как объявить функцию в C++. Напишите правило и назовите все элементы объявления.**

Чтобы объявить функцию в C++, необходимо указать ее имя, тип возвращаемого значения и параметры, которые она принимает. Пример объявления функции:

*int add(int a, int b);*

add — имя функции;

тип возвращаемого значения — int, и 2 целочисленных параметра a и b. Если функция не возвращает никакого значения, то тип тип указывается void

**2. Что такое прототип функции и как он используется компилятором?**

Прототип функции — это объявление функции, которое описывает ее имя, тип возвращаемого значения и параметры. Используется компилятором для проверки несоответствия типов данных и генерации кода для вызова функции.

Прототип функции обычно объявляется в начале файла C++, перед главной функцией или любой другой функцией, которая ее вызывает.

**3. Как объявить функцию, которая не возвращает никакого результата?**

Чтобы объявить функцию, которая не возвращает никакого результата, можно использовать ключевое слово void в качестве возвращаемого типа. Пример:

*void printHello() { cout << "Hello, world!" << endl; }*

**4. Как объявить функцию, которая может изменять свой параметр целого типа, например,  
 int x?**

для объявления функции, которая может изменять свой параметр целочисленного типа, необходимо передать параметр по ссылке. Пример:

*void change(int& x) { x = x + 1; }*

**5. Можно ли вызвать Swap, объявленную раньше, как Swap(x+y, x-y)? Поясните ваши соображения.**

Да, можно вызвать функцию Swap. Но данная функция не поменяет местами значения (Swap). Вместо этого в функцию Swap будут переданы выражения x+y и x-y в качестве параметров, над которыми можно будет выполнять действия внутри функции Swap. Чтобы поменять местами значения необходимо передать их по ссылке.

**6. Что возвращает функция c прототипом int &foo()?**

Функция int &foo() возвращает ссылку на целое число.

**7. Из каких типов файлов состоит модуль компиляции в C++?  
В чем назначение каждого из них?**

Модуль компиляции C++ состоит из нескольких типов файлов:

.cpp — файлы, содержащие исходный код C++, который необходимо скомпилировать.

.h — это файлы, содержащие объявления функций и классов, а также любые другие объявления, необходимые исходному коду. Заголовочные файлы включаются в исходные файлы с помощью директивы препроцессора #include.

.obj — это скомпилированные версии файлов исходного кода.

**8. Что является входом и выходом препроцессора? Когда он вызывается?**

На входе препроцессора находится исходный код, написанный на языке C++, а на выходе — версия исходного кода, готовая к компиляции. Препроцессор вызывается перед компилятором, и его цель — обработка исходного кода для включения заголовочных файлов и удаления комментариев.

**9. Как выполняется инструкция #include? Чем отличаются с точки зрения препроцессора:   
#include "utils.h" и  
#include <utils.h>?**

Инструкция #include выполняется препроцессором, который заменяет директиву содержимым указанного файла. Разница между *#include "utils.h"* и *#include <utils.h>* заключается в том, что первая сначала ищет файл в текущем каталоге, а вторая — в системных каталогах.

**10. Почему компилятор может компилировать вызовы функций, реализованных в другом модуле, если в cpp-файл включен соответствующий заголовочный файл? Поясните это с точки зрения действий препроцессора.**

Когда заголовочный файл включается в исходный файл, препроцессор копирует объявления из заголовочного файла в исходный файл. Это позволяет компилятору проверить правильность вызовов функций в исходном коде.

**11. Что возникает в результате компиляции? Как соотносятся результаты успешной компиляции с cpp-файлами проекта?**

Компиляция создает .obj-файлы из исходного кода, которые могут быть соединены для формирования исполняемой программы. Результаты успешной компиляции включают объектные файлы для каждого исходного файла в проекте. Эти файлы могут быть объединены в исполняемую программу, которая может быть выполнена компьютером.