Вопросы:

- ***Что такое Функция, отличие функции main() от других функций?***

Функция - подпрограмма, выполняющая какие-либо операции и возвращающая значение.

Функция — это набор инструкций, которые последовательно выполняются, для выполнения определенного задания.

Именно с первой инструкции в main() и начинается выполнение.

Когда программа выполняется, операционная система делает вызов функции main() и начинается её выполнение. Стейтменты в main() выполняются последовательно сверху вниз. В конце функция main() возвращает целочисленное значение (обычно 0) обратно в операционную систему. Поэтому main() объявляется как int main().

- ***Конструкция функции?***

Конструкция функции без возврата значения:

<**void**> <**имя функции**> (<**тип данных**> <**имя параметра**>, <**тип данных**> <**имя параметра**>, ...)

{ // блок стейтменов открываем

<**тело функции**>

} // блок стейтменов закрываем

Конструкция функции с возвратом значения:

<**тип данных**> <**имя функции**> (<**тип данных**> <**имя параметра**>, <**тип данных**> <**имя параметра**>, ...)

{// блок стейтменов открываем

<**тело функции**>

return < **переменная типа данных функции**>;

} // блок стейтменов закрываем

- ***Что такое прототип функции зачем он нужен?***

Предварительное объявление сообщает компилятору о существовании идентификатора **ДО** его фактического определения. В этом случае перед вызовом функции надо ее дополнительно объявить. Объявление функции еще называют прототипом.

- ***Что такое определение функции?***

Первая строка представляет заголовок функции. Вначале указывается возвращаемый тип функции. Если функция не возвращает никакого значения, то используется тип void. Затем идет имя функции, которое представляет произвольный идентификатор. К именованию функции

применяются те же правила, что и к именованию переменных. После имени функции в скобках идет перечисление параметров. Функция может не иметь параметров, в этом случае указываются пустые скобки. После заголовка функции в фигурных скобках идет тело функции, которое содержит выполняемые инструкции. Для возвращения результата функция применяет оператор return. Если функция имеет в качестве возвращаемого типа любой тип, кроме void, то она должна обязательно с помощью оператора return возвращать какое-либо значение.

- ***Что такое функции с возвращаемым значением?***

Тип возвращаемого значения (или «тип возврата») - он указывается при объявлении функции, перед её именем. Тип возврата не указывает, какое именно значение будет возвращаться. Он указывает только тип этого значения. Затем, внутри вызываемой функции, мы используем оператор **return**, чтобы указать возвращаемое значение — какое именно значение будет возвращаться обратно в **caller**.

- ***Что такое функции без возвращаемого значения?***

Функция имеет тип возврата **void**, который означает, что функция не возвращает значения. Поскольку значение не возвращается, то и оператор **return** не требуется.

- ***Перегрузка функций, что допускается перегружать?***

Перегрузка функций — это особенность C++, которая позволяет создать несколько функций с одним и тем же именем, но с разными параметрами. Язык C++ позволяет определять функции с одним и тем же именем, но разным набором параметров.

Чтобы определить несколько различных версий функции с одним и тем же именем, все этиверсии должны отличаться как минимум по одному из следующих признаков:

- имеют разное количество параметров;
- соответствующие параметры имеют разный тип.

- ***Параметры по умолчанию в функции?***

Параметр по умолчанию (также «необязательный параметр» или «аргумент по умолчанию») — это параметр функции, который имеет определенное значение по умолчанию.

- ***Что значик ключевое слово inline?***

Ключевое слово **inline** используется для запроса, чтобы компилятор рассматривал вашу функцию как встроенную.

- ***Допускается ли в C++11 вложенность функций?***

В языке C++ одни функции не могут быть объявлены внутри других функций.

- ***Ключевое слово auto что это такое? Что допускается использовать с авто с функциями***

Ключевое слово **auto** использовалось для явного указания, что переменная должна иметь автоматическую продолжительность: *auto int boo(7);*

Т.к. все переменные в современном C++ по умолчанию имеют автоматическую продолжительность, и, если не прописывать самому другой тип продолжительности, ключевое слово auto стало излишним и, следовательно, устаревшим.

- ***в C++ 11?***

Начиная с C++ 11, ключевое слово **auto** делает, при инициализации переменной оно может использоваться вместо типа переменной, чтобы сообщить компилятору, что он должен присвоить тип переменной исходя из инициализируемого значения.

- ***trailing функции в C++ 11?***

Синтаксис типа возвращаемого значения **trailing**, когда компилятор делает выводы о типе возвращаемого значения после остальной части прототипа функции.

- ***Плюсы использования функций.***

Структура. Функция — это как мини-программа, которую мы можем записать отдельно от головной программы, не заморачиваясь при этом об остальной части кода. Это позволяет разбивать сложные задачи на более мелкие и простые, что кардинально снижает общую сложность программы.

Повторное использование функций. Одну и ту же функцию можно вызывать несколько раз, что очень полезно.

Тестирование. Поскольку функции убирают лишний код, то тестировать кода остается меньше.

Модернизация. Когда нужно внести изменения в программу или расширить её функционал — функции отличный вариант.

Абстракция. Для того, чтобы использовать функцию, нам нужно знать её имя, данные ввода, данные вывода и где эта функция находится. Нам не нужно знать, как она работает.

- ***Что такое область видимости, что такое локальная, глобальная область видимости?***

Область видимости определяет, где можно использовать переменную.

Параметры функции, и переменные, которые объявляются внутри тела функции, имеют локальную область видимости (local scope).

Глобальная область видимости весь файл.

- ***Что такое конфликт имен?***

Если два идентификатора находятся в одной и той же программе таким образом, что компилятор не может различить их, он выдаст ошибку. Эта ошибка, как правило, называется конфликтом имен (или столкновением имен).

- ***Что такое оператор разрешения контекста, или оператора области видимости?***

Оператор разрешения контекста (оператора области видимости (::)) имеет самый высокий приоритет и применяется в двух формах:

- унарная — ссылается на внешний контекст;
- бинарная — ссылается на контекст класса.

- ***Что такое блок стейтментов, зачем он нужен?***

Блок стейтментов(или так называемые составные операторы) – это группа стейтментов, которые обрабатываются компилятором как одна инструкция. Блок начинается с символа { и заканчивается символом }, стейтменты находятся внутри.

- ***Глобальные (не константные) переменные, глобальная область видимости?***

Глобальные (неконстантные) переменные – это те, которые объявлены вне блока. Они имеют статическую продолжительность, что означает, что они создаются при запуске программы и уничтожаются, когда программа завершает свое выполнение. Глобальные переменные имеют файловую область видимости (или неформально “глобальную”), то есть их можно использовать в любом месте файла, в котором они объявлены.

Объясните что выведет код и исправен ли он, и как исправить чтоб работал:

```
// задание 1
#include <iostream>
int return 5(){ // убрать пробел должно быть: return5
return 5;
}
int return8(){
return; // добавить значение должно быть: return8
}
int main(){

std::cout << (return5() + return8() ) << std::endl;
return 0 // завершить стейтмент должно быть: return 0;
}
```

Выведет сумму двух целых чисел // 5+8 = 13.

```
// задание 2
// отсутствует библиотека ввода/вывода
int return() { // неправильное именование функции зарезервированным словом
    return 5;
    int return() // вложенность функций не допускается
    {
        int k ;
        k{3}; // не верное присвоение значение переменной
        return k;
    }
}
int main(){
    std::cout << return() << std::endl; // конфликт имен
    std::cout << return() << std::endl;
    return 0;
}

// Исправленный код

#include <iostream>
```

```
int return3()
{
    int k{3} ;
    return k;
}

int return5() {
    return 5;
}
int main(){
    std::cout << return3() << std::endl;
    std::cout << return5() << std::endl;
    return 0;
}
```
Выведет два целых чисела: 3 и 5.

```
// задание 3
#include <iostream>
void prints
()
{

    std::cout << 'O_o' << std::endl;
}
int main ()
{
    /* не правильная запись вывода
так как вызывается функция типа void в которой уже имеется вывод значений
*/
    std::cout << prints() << std::endl;
    return 0;
}
// Исправленный код
#include <iostream>
void prints
()
{

    std::cout << 'O_o' << std::endl;
}
int main ()
{

    prints();
    return 0;
}
```
Выведет мусор.

```
// задание 4
#include <iostream>
/* функция выводит только одно значение при помощи return
int getNumbers()
{
    return 6;
    return 8;
}
int main()
{
    std::cout << getNumbers() << std::endl;
    std::cout << getNumbers() << std::endl;
    std::cout << getNumbers << std::endl; // переменной getNumbers не существует
    return 0;
```

```
}
// Исправленный код
#include <iostream>
int getNumbers()
{
    return 8;
}
int getNumbers1()
{
    return 6;
}
int main()
{
    std::cout << getNumbers() << std::endl;
    std::cout << getNumbers1() << std::endl;
    return 0;
}
```
Выведет целые числа 8 и 6.

```
// задание 5
int main()
{
    std::cout << multiply(7, 8) << std::endl;
    return 0;
}
// функция определена ниже вызова
// в функции один аргумент
void multiply(int a)
{
    // инструкция не закрыта символом (;)
    return a * b
}

// Исправленный код
#include <iostream>
void multiply(int a, int b);
int main()
{

    multiply(7, 8);
    return 0;
}
void multiply(int a, int b)
{
    //return a * b;
    std::cout << a * b << std::endl;
}
```
Выведет произведение 7*8=56.

```
//Чтобы программа скомпилилась нужно что-то добавить а нельзя удалять и
комментировать:
// 6
1 #include <iostream>
2 int tmp = 1;
3 int getNumbers( {
4 int tmp = 1;
5 return 8;
6 }
7 int main()
8 {
9 int tmp = 3;
10
```

```
11 std::cout << tmp << "Чтобы программа скомпилилась нужно что-то добавить а нельзя
удалять
12
13 и
14
15 комментировать "
16 << std::endl;
17
18 return 0;
```

В конец строки 11 добавить символ (\) для переноса комментария на новую строку

В конец строки 13 добавить символ (\) для переноса комментария на новую строку