1. В чём заключается концепция встраивания вызовов функции.

Используя ключевое слово inline компилятор вставляет тело функции в место её вызова подходящим образом. Преимущество у этого прежде всего для небольших функций 1-5 строк, затрачивается меньше системных средств на переход из одного места кода в другое, на создание новых переменных и иногда копирование, все операции функции выполняются непосредственно в месте вызова.

2. Какие аргументы функции могут иметь значение по умолчанию?

Аргументы со значением по умолчанию все должны быть в конце. В последовательности таких аргументов не должно быть перерыва.

Например, так правильно:

```
Int f(int x, int y = 0, int z = 1, int m = 32);
```

А вот так вот нет:

```
Int f(int x = 0, int y, int z, int m = 32);

Int f(int x, int y = 0, int z = 1, int m);
```

3. На основании чего разрешается выбор перегруженной функции?

На основании типа возвращаемого значения, количества аргументов и типов аргументов.

4. Как обеспечить состояние в функциях и лямбда-выражениях?

Если я правильно понял вопрос, то это про то, как мы можем передать сюда значения функций или переменные.

Можно использовать глобальные переменные. Кодовое слово static, чтобы переменная, созданная в какой-то узкой области жизни, хранилась в статической памяти, а не в стеке, и следовательно не удалялась после выхода из области жизни. Вообще есть способы передачи переменных по значению, по ссылке, по указателю. Если функция должна быть аргументом другой функции, то либо мы пользуемся указателем void(*ptr)(int), void - тип возвращаемого значения, ptr - название функции, int - аргументы функции. Либо библиотекой functional и там просто std::function<void(int)> ptr; ptr, void и int здесь подразумевают то же самое, что и в рассмотренной ситуации через указатели.

5. По каким причинам макросы считаются опасным инструментом?

Макросы нельзя отлаживать. У них нет пространства имен. Могут делать что-то непонятное, при неверном задании и это сложно проверить иногда.