Книги:

0) Прата С++
1) Майерс Скот Эффективный С++ (подразумевает хорошие знания по С++)
2) github.com

Северов - лекции по архитектуре компьютеров

Language = C++ ; // сначала присваивает значение, а потом уже увеличивает значение,
//постинкримент.

Крайне низкоуровневый (ассемблер с человеческим лицом)
Переизобретаем* велосипед.

Заказчикам зачастую поставляются лишь хидера и библиотеки уже в скопленном виде, этого особо не модифицируешь.

Что мы имеем в С из-за С++: const, // ;

Из-за сложности языка появилось много утилит, анализирующих код, в частности класс этих утилит назывался статический аналищатор, они смотрели исходный код на С++ и подчеркивали некоторые странные места (где безнаковой переменной присваиваем знаковое значение, например). Не только помогут увидеть, но и исправить, поправить код.

Основные принципы ООП:

1) Полиморфизм (способность функции обрабатывать данные разных типов);
2) Инкапсуляция (все что можно скрыть должно быть скрыто);
3) Наследование (идея в простой эволюции, машина -> полицейская машина);

Как, например, R^n и метрические пространства. Метрическое пространство включает в себя Евклидово пространство и R^n. Это и отражает пример наследования.

Сегодня на паре мы знакомимся с полиморфизмом.

Сигнатура - типы параметров функции, так что можно создать: (и их количество)

```
void swap (int* a, int* b);
void swap (double* a, double* b);
```

Но нельзя создать:

```
int swap (int* a, int*b);
doble swap();
```

Соответсвенно, они должны различаться сигнатурой.
Шаблоны - особенность языка С++ (в отличие от С, заимствовали из какого-то другого языка);

Template - шаблон: template <typename T> void swap(T, *a, T *b) if

Для своих типов данных можно задать явную специализацию. Она будет говорить программе не генерировать новую функцию под определенный тип, а использовать специализированную.

*передает по ссылке
& - ссылка (значение может быть присвоено лишь один раз)
* - указатель (можно несколько раз)
См. программу. И разберись, т.к. не понятно.

В языке С++ можно отдельно описать операции по типу сложения для каждого типа данных. То есть что-то она совращает. (возвращает)

decltype(a + b) c = (a + b); // добавлен в С++11, присвоит с тип значения суммы (а + б);

auto //с момента создания языка для создания локальных переменных, теперь для переменной определения типа

Int a = 0;
auto int = 0; // для стека
static //глобальная переменная
register //

//В С++ никто не пользовался auto, т.к. она не нужна.
//ВПЕРВЫЕ было переопределение в стандарте.

auto - определяет значение переменной

auto c = a;

Оценивание: result() + 1 (+1 только при достижении максимума в result);
Нужно вызвать восхищение.

Итог знакомства:

Мы прошли полиморфизм, явную специализацию.
Не для всех структур и типов данных заданы нормальные значения которыми можно инициализировать.
Не для всех специфичных типов задан шаблон операции.
Есть как минимум операция присваивания, сравнения, да все что угодно.

Познакомились как определять типы выражений и переменных, что автоматически вычислялось. Как определять типы возвращаемых значения у шаблонных функций.

Рассмотрим еще интересные примеры перегрузки операции.

```
int Sum(int a, int b) // defines what to summon by number of vatiables
{
}

int Sum(int a, int b, int c)
{
}
```

В С и С++ есть возможность создать функцию с переменным числом переменных.

Добавили inline для функций в С++; Теперь функция просто копируется в мейн без вызова (без call); *В стек, в стэк, call, еще кое-что, освобождение и продолжение/ inline быстрее.