Второе занятие по программированию (11.02.2019)

Отладка программы:

На Github на нас натравили статический анализатор.

Создание проверки для нашей программы.

Например, нужно всегда ставить фигурные скобки у условного оператора. Потому что это будет всегда верно, иначе можно ошибиться.

Мы с вами пользуемся Git'ом. Системой контроля версий. Таких было несколько, до него была, например: cvc, svn, mircurial, git, … . Основным стандартом сейчас считается git.
SVN - проще, его проще освоить и показать примеры. Сейчас доживает последние дни. В старых проектах может быть SVN.

Система контроля версии помнит предыдущее состояние.
У системы есть история и она запоминается состояния, чтобы возвращаться к прошлому коду.
Исторически они появились чтобы осуществлять совместную работу. (хоть со всего света)
Отдельные исправления всех людей может соединять воедино. (детище Линукса Торвальдс)

git branch max_hw - задает новую ветку ;
git checkout max_hw - переходим в нашу часть ;

git pull - (git fetch - берет все изменения в оригинале) + (get merge - соединить) ;
git rebase master

Когда мы создали или поменяли файл, выполняется операция:
git add - добавить
git commit -m - запоминай текущий файл

git push - изменения в оригинальном репрозитории
git push origin max_hw - в мастер

Области видимости:

int a = 0; int main(){ int a = 1; { *int a = 2;* } return 0; } - фигурные скобки;

static - идея в том, что у нас может быть куча функций с одинаковым названием. Мы можем поставить static и тогда они будут видны лишь в данном файле.

Ключевое слово namespace:

Есть команда: using namespace Test; //будет использовать определенное простр. имен
Осторожней с этой командой. Т.к. В настоящем программировании это может вызвать проблемы.

Лучше явно говорить, где мы ожидаем найти функцию.

Эта команда вызывает ту функцию, которую найдет компилятор, а это не всегда то, чего вы ожидаете найти.

Почему С не устраивал людей? Перейдем к классам.

Классы:

- Полиморфизм (0 наш урок)
- Инкапсуляция (1 урок, namespace, class)
- Наследование

В классах есть еще один метод инкапсуляции. Он позволяет скрыть что-то внутри класса.
И вне класса это видно не будет.

Каждый наш метод смены поля в С мы использовали указатели.
У каждого метода нашего класса есть неявный парамент, который выглядит как указатель:
simpleClass::simpleClass(/*simpleClass* a*/
this -> a

Вот хотим мы сравнить два числа на плоскости, какая лежит ближе к определенной точке.
Возвращаем указатель на эту точку.

Классы string:

Стандартные алгоритмы подразумевают, что строка оканчивается нулем.

Для типов классов можно создавать свои операции.
Функций у стандартной строки много:

Механизм friends позволяет другим классам залезать в в ваш private!

friend ostream operator << (String & str);