Constructors

Used to reset an object's member variables when the object is first created; otherwise random

Example: Gassy People

```
class Gassy
{
    public:
        Gassy(int age, bool ateBeans)
        {
            m_age = age;
            m_ateBeans = ateBeans;
        }

        int getFartsPerHr()
        {
            if (m_ateBeans == true)
            {
                return 100;
            }
            return (3 * m_age);
        }
    private:
        int m_age;
        bool m_ateBeans;
};
```

- If a constructor has parameters -> **must pass values**

```
int main()
{
    Gassy betty(18, true); //OK
    Gassy alan; //NOT OK
}
```

Order of Construction

- Data members are always initialized first
    - Adds code to outer class's constructor to first call the **default** constructors of all member objects

```
HungryNerd()
    Call myBelly's default constructor
    Call myBrain's default constructor
```

```
{
    myBelly.eat();
    myBrain.think();
}
```

- After type data members are constructed...
    - C++ executes body of outer object's constructor

1. --
2. Construct data members
    - consult the member initialization list
    - if a data member is not listed in the initializationlist:
        - if builtin type: left uninitialized (theninitialize in body)
        - if class types: the class's default constructoris called
3. Execute the body of the constructor

When Constructors Are Called...

- Called any time you create a **new variable** of a class
- Called *N* times when you create an array of size *N*
- Called when you use `new` to dynamically allocate a new variable
- If variable declared in a loop -> newly constructed during each iteration
- **NOT CALLED** when you define a pointer variable to this type

Constructor Overloading & Default Parameters

- Class can have many different constructos
    - Each *must* have different parameters and/or types

```
Gassy()
{
    m_age = 0;
    m_ateBeans = false;
}
```

    - Now we can pass no parameters
- Implicit default constructor: if you don't create ANY constructor, C++ generates one for you
    - Does NOT initializes primitive member variables

```
Gassy()
{}
```

```
String::String(const char* value = "")
{
    ...
}
```

- If a constructor *can be called with no arguments*, then it is a default constructor
- Only specific in declaration, not implementation

```
void fart(int length = 10, int volume = 50)
{
    ...
}

int main()
{
    fart(20, 5);
    fart(5);
    fart();
    fart(, 30); //ILLEGAL
}
```

> If the *j*th parameter has a default value, then ALL of the following parameters (*j+1* to *n*) must also have default values

Member Initialization List

> Required when a class contains objects without a default constructor

- Auto-generated C++ code always calls the default constructor of member variables (takes NO parameters)
    - If you have a defined constructor -> will not be called -> BAD NEWS!
    - **Compilation error**
- Initialization list: code that explicitly calls member variable constructors WITH a parameter

```
Stomach(int startGas)
{
    myGas = startGas;
}
```

```
HungryNerd()
    : myBelly(10)
    Call myBrain's default constructor
{
    myBelly.eat();
    myBrain.think();
}
```

Initializer List Accepts Constants & Variables

```
HungryNerd(int startingGas)
    : myBelly(startingGas)
    Call myBrain's default constructor
{
    myBelly.eat();
    myBrain.think();
}
```

Purpose of Initialize List

- **Performance**: initialize primitive variables, string m_name` is a class type
    - if initialize in body, default constructor called irst and initializes it to empty string, then nitialized in body --> WASTE
    - instead: use initializer list
- **Mandatory**: head of stick figure won't compile if nitialize everything in body: **the circle has no default onstructor**