CSCI 112
Programming with C

C++ Intro

Getting Ready For Hello World

- I/O differences
  - #include <iostream>
    using namespace std;
  - cout << “Hello World\n”;
  - OR cout << “Hello World” << endl;
- Naming conventions for source files
  - I will use .cpp
  - You can use .C, .cc also – it doesn’t matter to the compiler
- Compiler
  - g++
  - g++ -o hello –Wall hello.C

Input/Output Streams

- #include <iostream>
- using namespace std; // removes need for std::
  - differentiates between identifiers of the same name
  - keeps us from having std:: for every write or read
  - std means that cin and cout are found in the standard library namespace
    - a namespace just is a way of grouping functions, etc
- cin >>
  - for reading in input from the console
- cout <<
  - for writing output to the screen

Procedural Programming VS Object Oriented Programming

- Object Oriented programming – C++
  - Develop classes – an object is an instance of a class
    - Contain Data
    - Run procedures known as methods
    - Can change state/mode
- Procedural programming – C or can use C++
  - Develop functions
    - computations
      - expressions
    - Do one small part of a bigger job
  - Logic
  - Doesn't change state

Procedural Programming VS Object Oriented Programming

- Difference is how you approach the problem you are trying to solve
- In all programs, there are two primary components: the data (the stuff a program knows) and the behaviors (the stuff a program can do to/with that data). OOP says that bringing together data and its associated behavior in a single location (called an “object”) makes it easier to understand how a program works. FP says that data and behavior are distinctively different things and should be kept separate for clarity.

From https://www.codenewbie.org/blogs/object-oriented-programming-vs-functional-programming

Procedural Programming VS Object Oriented Programming

- Example: Give everyone in your organization a raise
- OOP:
  - Class for an employee, 3 attributes – name, id, salary
  - Now we create an object for each employee (pulling the info from a database)
  - Use a method to increase his salary
  - Use a method to update the database for that object
- PP:
  - Loop until end of data
  - Create a function to read one set of data from the database
  - This data includes the name, id and salary of the employee
  - Create another function to that increases the salary
  - Create another function to update that data element into the database

From https://www.codenewbie.org/blogs/object-oriented-programming-vs-functional-programming

Why Learn C++

- Developed by Bjarne Stroustrup at Bell labs 1979-1984
- Applications written in C++:
  - Many of the Adobe tools – Adobe Photoshop
  - Games
  - 3D animation, modeling, simulation, rendering software
  - Google Chrome, and Mozilla Internet browser Firefox
  - MySQL
  - Bloomberg RDBMS
  - Winamp Media Player
  - Compilers for C# and Java
  - Microsoft Office
  - to name a few

Why Learn C++

- C++ is known to be a very powerful language. C++ allows you to have a lot of control as to how you use computer resources, so in the right hands its speed and ability to cheaply use resources should be able to surpass other languages. Thanks to C++'s performance, it is often used to develop game engines, games, and desktop apps. Many AAA title video games are built with C++.

Taken from www.bestprogrammlanguagefor.me/why-learn-c-plus-plus

Why Learn C++

- Link: https://blog.appdynamics.com/engineering/the-bedrock-of-the-software-world-cpp-programming-language/

Object Oriented Programming

- Class
  - has attributes (data)
  - has behavior (functions – called methods)
    - special method – constructor – called whenever a new object is created
    - usually use set and get methods for accessing data
- Object
  - instance of a class

Classes

- think of it as a blueprint for creating an object
- in c terms – the class is the data type (struct) and the object is the variable of that type.
- In C the unit of programming is the function – data supports the actions that the functions are to perform
- In C++ the unit of programming is the class from which objects are instantiated
- think of it as nouns (C++) vs verbs (C – functions)

Comes down to how you design your program

Procedure types

- Methods (embedded in classes)
- Functions (outside of classes)
- the main function

Car Simulation Example

C

```
typedef struct {
        char* color;
        char* model;
        int id;
        char fuel_type;
} Car;

Car myford;
// Functions start, brake, accelerate can
be called from anywhere and you must
pass it the variable of type car
strcpy(myford.color, “blue”);
start(myford);
```

Car Simulation Example

C++

```
Class Car {
public:
        Car(string, string, long int, char);    //
constructor
        void start();
        void accelerate();
        void brake();
private:
        string color;
        string model;
        long int id;
        char fuel_type;
}

Car myford(“blue”, …);    // will call constructor
…
myford.start();
```

C

```
typedef struct {
        char* color;
        char* model;
        int id;
        char fuel_type;
} Car;

Car myford;
// Functions start, brake, accelerate can be
called from anywhere and you must pass it
the variable of type car
strcpy(myford.color, “blue”);
…
start(myford);
```

Object Oriented Programming

- Data hiding
  - Data Encapsulation

Is the mechanism whereby the implementation details of a class are kept hidden from the user.

The user can only perform a restricted set of operations on the hidden members of the class by executing special functions commonly called *methods*.

keywords: public, private, protected

Benefit: details of class changes but interface stays the same

From http://www.cs.mun.ca/~donald/bsc/node13.html

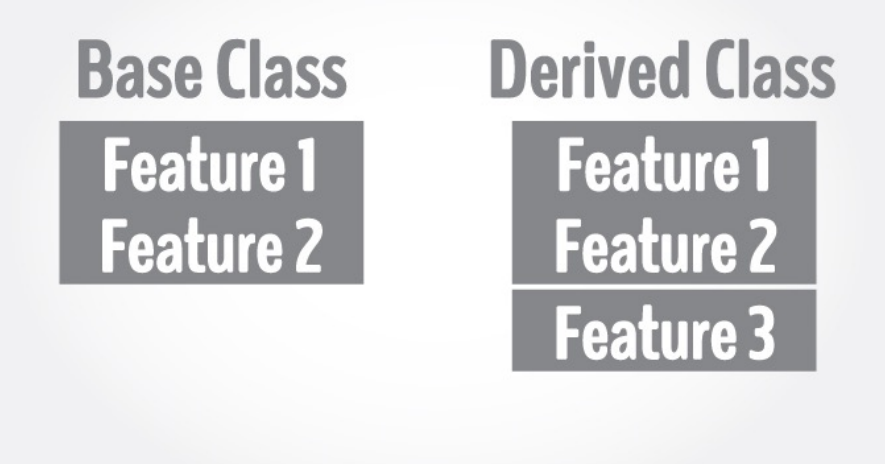
Object Oriented Programming

- Abstraction
  - focus on what the object does instead of how it is done
    - Details inside class
    - High level outside class
    - Easier to maintain
    - Can create the classes first or last – apart from the program
    - Reuse
  - Represent the essential feature without detailing the background implementation or internal working detail
  - Focus on only those things that matter our module.
    - Modifying one independent module does not impact the other modules.
    - The only knowledge one needs to know is what a module gives you.
    - The caller of that module does not need to bother about how the task is achieved or what exactly is happening in the background.

From https://dev.to/charanrajgolla/beginners-guide---object-oriented-programming

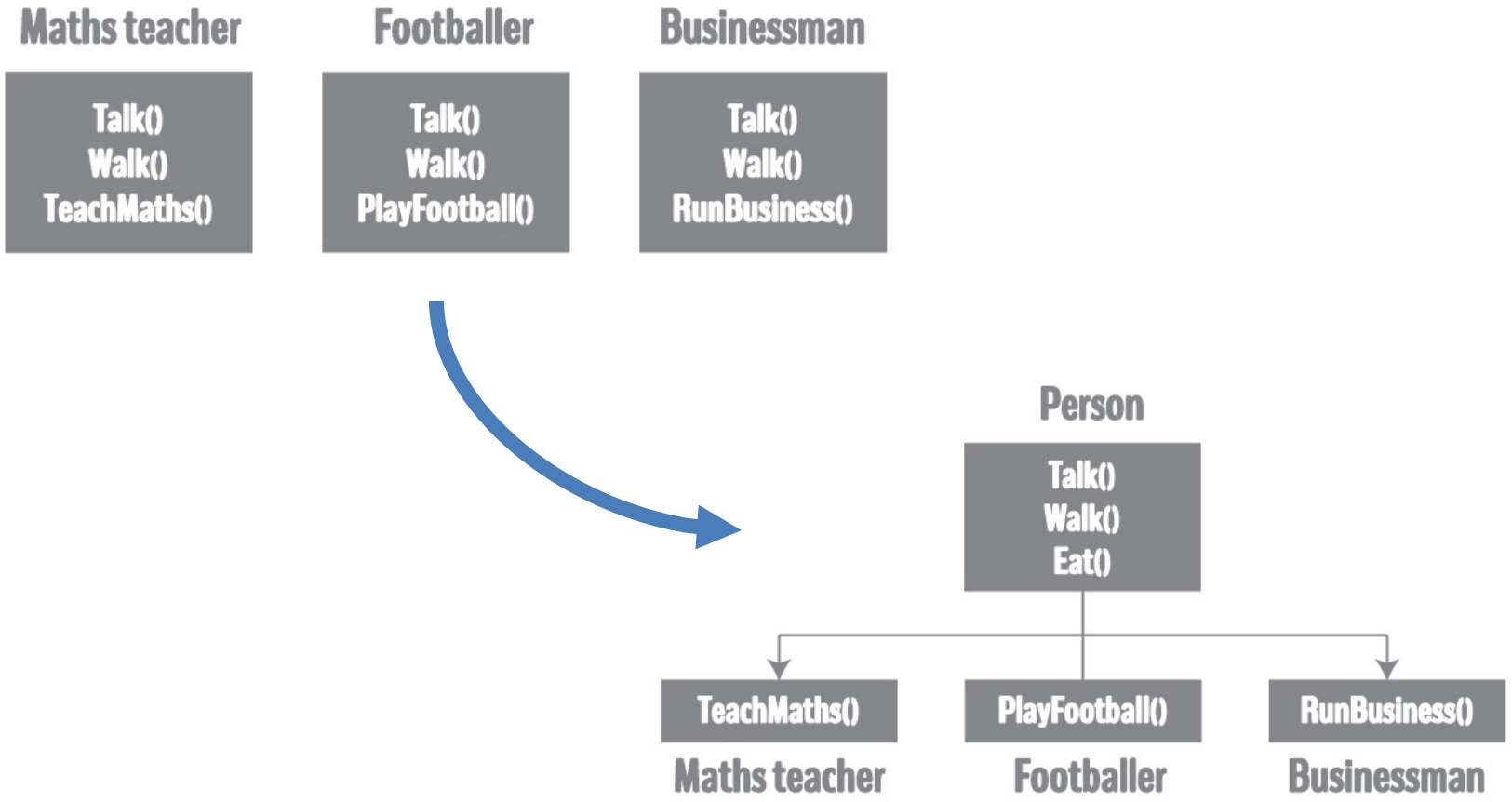
Object Oriented Programming

- Inheritance
  - Key feature of object oriented programming
  - It allows creating a new class (derived class) from an existing class(base class).
  - The derived class inherits all the features from the base class and can have additional features of its own.



From https://www.programiz.com/cpp-programming/inheritance

Inheritance Example



Object Oriented Programming

- Polymorphism
  - Is the concept that there can be many different implementations of an executable unit and the difference happen all behind the scene without the caller awareness.
  - A member function will cause a different function to be executed depending on the type of object that invokes the function.
    - For example, we could have two functions called start for Car
      - start(int);
      - start();
    - Works great for constructors
      - Car car1;
      - Car car2(model, …);

From https://dev.to/charanrajgolla/beginners-guide---object-oriented-programming

Things to Watch Out For

- if you use gcc instead of g++, you will get weird errors
- if you do: cout >> instead of cout << you will get a ton of weird errors