Madeline Endres

CLASSES

Classes

- So far we have seen simple data types eg: int, bool, char, double
- We have also seen arrays, or lists of like data types stored together in memory
- What if we want to model real life objects by combining different types into one big type? Time for classes!!!!

When would we use Classes?

- A class groups data together
- By default, everything in a class is private

- Used when you want to model objects that have member variables and perform functions

Class Syntax

```
class Name {
   private:
   //Member variables associated with the class
   //Functions used only in other member functions

   public:
   //Member functions that modify and do
   calculations on the member variables

};  //Don't forget the semicolon!
```

There are two options of where to implement the the member functions of a class!

Public vs. Private

- public: anyone in any file can modify
- private: only other things in the class can modify.

- Keep your data private!

Option 1: One File:

```
class Name {
   private:
   //Member variables associated with the class
       string name;
       int age;

   public:
   //Constructor
       Name(arguments) { }
       void setAge(int a) {
              age = a;
       }
};  //Don’t forget the semicolon!
```

Option 2: Separate Files

Name.h

```
class Name {
   private:
   //Member variables
   associated with the
   class
   string name;
   int age;

   public:
   //Prototypes of functions
       Name(arguments);
       void setAge(int a);
};  //Don't forget the
   semicolon!
```

Name.cpp

```
//implementations of member
    functions of the class

//Constructor -> note the scope
    resolution operator ::
Name::Name() { };
void Name::setAge(int a) {
        age = a;
}
```

Constructors:

- The member variables and functions of a class act like a general blueprint of a house
- When you actually make an instance of a class, it is like you are building a specific house.
- Constructors are what allow you to get from the blue print to the actual house
  - Kind of like a contractor on a construction site
  - All classes need a constructor
    - If you don't write one, the compiler makes a default

Constructor syntax

Default Constructor

```
ClassName() {

//Either empty or give
  member variables
  default values

}

//used: ClassName id;
//or ClassName id();
```

Custom Constructor

```
ClassName(type name1,
  type name2, ...) {
     //Initialize
  variables to passed in
  parameters

}

//Used: ClassName
  id(name1, name2,....);
```

Overloading Constructors:

- You can have multiple constructors!!!!!
- Think of this like having several different starting blueprints for houses in a subdivision
  - They are all very simular, but construction starts with different parameters
    - eg: number or bedrooms, paint color

- Note: If you supply one custom constructor, the compiler no longer supplies a default

Constructor overloading examples:

If you have the following constructors implemented, which one is called?

```
University();

University(int age,
   string name);

University(string state,
   int stuBodysize);

University(bool public,
   string &name);
```

What happens here if the top constructor is called?

```
School(string name1,
   string state1) {
   name = name1;
   School(state1);
}
School(string state1) {
   state = state1;
}
//In main:  School
   UAA("UAA", "Alaska");
```

Which are valid constructors?

```
Sun() {
}

Moon Moon() {
  return this.Moon;
}

Day(Month month,
  Year year) {
}
```

```
Student (Student stu) {
  age= stu.age;
  name = stu.name;
  year = stu.year;
  return;
}

void Corgi() {
}
```

Classes Example: Prototypes in .h file

```
class MichiganStudent {
    private:
        string name;
        int UMID;
        string courses[15];
        int numCourses;
        string major;
        double GPA;

    public:
        //Default constructor
        MichiganStudent();
        //Custom Constructor
        MichiganStudent(string
                studentName, int ID,
                int CoursesNum, string,
                stuMajor, double
                grades);
        //Prints student
        void printStudent();
        //Changes the major of the
        //          student
        void changeMajors(string
                newMajor);
};
```

Getter and Setter Functions

- A great way to deal with private data:

```
void setName(string name1) {
   name = name1;
}//Setter Function

string getName() {
   return name;
}//Getter function
```

Why use getter and setter functions?

triangle.h

```
class Triangle()  {
  public:
      double, a, b, c;
      Triangle();
      //returns area
      //   of triangle
      double area();
};
```

main.cpp

```
int main() {
  Triangle tri;
  tri.a = 3;
  tri.b = 4;
  tri.c = 8;
  cout << tri.area();
}
```

//What's the problem?

Use a setter!!

triangle.h

```
class Triangle()  {
  private:
     double, a, b, c;
  public:
      Triangle();
     double area();
     setSides(double a,
       double b, double c);
};
```

main.cpp

```
int main() {
  Triangle tri;
  tri.a = 3;
  tri.b = 4;
  tri.c = 8;
  cout << tri.area();
}
```

Use a setter!

```
void Triangle::setSides(double a,
  double b, double c) {



}
```