CSCI 390 – Special Topics in C++

Lecture 11

9/25/18

Time To Turn Off Cell Phones

C++ Is Object Oriented

- Implemented by **`struct`** / **`class`** / **`union`**.
- Creates a user defined type.
- **`struct`** is usually most appropriate, but **`class`** most often used. **`union`** is rarely used.
- Only minor differences.
- Syntax:
  (**`struct`** | **`class`** | **`union`**) <id> [**`:`**<hdr>]
  `{`
  <object body>
  `};`

C++ Objects
Minimal Example

```
#include <iostream>
#include "object.h"

int main(int argc, char *argv[])
{
  sMinEx StructMinEx;
  cMinEx ClassMinEx;
  uMinEx UnionMinEx;

  return 0;
}
```

```
...Program finished with exit code 0
Press ENTER to exit console.
```

```
// object.h
struct sMinEx
{
};

class cMinEx
{
};

union uMinEx
{
};
```

C++ Objects
Member Variables

```
#include <iostream>
#include "object.h"

int main(int argc, char *argv[])
{
  sPoint Point;

  return 0;
}
```

```
...Program finished with exit code 0
Press ENTER to exit console.
```

```
// object.h
struct sPoint
{
  double x;
  double y;
};
```

C++ Objects
Constructors / Destructors

```
#include <iostream>
#include "object.h"

int main(int argc, char *argv[])
{
  sPoint Point;

  return 0;
}
```

```
...Program finished with exit code 0
Press ENTER to exit console.
```

```
// object.h
struct sPoint
{
  sPoint(void) { return; }
  virtual ~sPoint(void) { return; }

  double x;
  double y;
};
```

C++ Objects Automatically Execute

```
#include <iostream>
#include "object.h"

int main(int argc, char *argv[])
{
  sPoint Point;

  return 0;
}
```

```
Hello sPoint(void)
Goodbye ~sPoint(void)


...Program finished with exit code 0
Press ENTER to exit console.
```

```
// object.h
#include <iostream>
struct sPoint
{
  sPoint(void) {
    std::cout << "Hello sPoint(void)" <<
      std::endl;
    return;
    }
  virtual ~sPoint(void) {
    std::cout << "Goodbye ~sPoint(void)" <<
      std::endl;
    return;
    }

  double x;
  double y;
};
```

C++ Objects
Default Initialization

```
#include <iostream>
#include "object.h"

int main(int argc, char *argv[])
{
  sPoint Point;

  return 0;
}
```

```
Hello sPoint(void)
Goodbye ~sPoint(void)

...Program finished with exit code 0
Press ENTER to exit console.
```

```
// object.h
#include <iostream>
struct sPoint
{
  sPoint(void) : x(0.0), y(0.0) {
    std::cout << "Hello sPoint(void)" <<
      std::endl;
    return;
    }
  virtual ~sPoint(void) {
    std::cout << "Goodbye `sPoint(void)" <<
      std::endl;
    return;
    }

  double x;
  double y;
};
```

C++ Objects
Constructor Overload

```
#include <iostream>
#include "object.h"

int main(int argc, char *argv[])
{
  sPoint Point;
  sPoint(1.0, 1.0);

  return 0;
}
```

```
sPoint(void)
sPoint(double, double)
~sPoint(void)
~sPoint(void)

...Program finished with exit code 0
Press ENTER to exit console.
```

```
// object.h
#include <iostream>
struct sPoint
{
  sPoint(void) : x(0.0), y(0.0) {
    std::cout << "sPoint(void)\n"; return; }
  sPoint(double _x, double _y) : x(_x), y(_y) {
    std::cout << "sPoint(double, double)\n";
    return; }
  virtual ~sPoint(void) {
    std::cout << "~sPoint(void)\n"; return; }

  double x;
  double y;
};
```

C++ Objects
Constructor Initialization

```
#include <iostream>
#include "object.h"

int main(int argc, char *argv[])
{
  sPoint Point{-1.0, -1.0};
  Spoint Point2(1.0, 1.0);

  return 0;
}
```

```
sPoint(double, double)
sPoint(double, double)
~sPoint(void)
~sPoint(void)

...Program finished with exit code 0
 Press ENTER to exit console.
```

```
// object.h
#include <iostream>
struct sPoint
{
  sPoint(void) : x(0.0), y(0.0) {
    std::cout << "sPoint(void)\n"; return; }
  sPoint(double _x, double _y) : x(_x), y(_y) {
    std::cout << "sPoint(double, double)\n";
    return; }
  virtual ~sPoint(void) {
    std::cout << "~sPoint(void)\n"; return; }

  double x;
  double y;
};
```

C++ Objects
"." Accesses Object Members

```
#include <iostream>
#include "object.h"

int main(int argc, char *argv[])
{
  sPoint Point{-1.0, -1.0};
  sPoint Point2{x:1.0, y:1.0};

  std::cout << "Point: (" << Point.x << ", "
    << Point.y << ")\n";
  std::cout << "Point2: (" << Point2.x << ", "
    << Point2.y << ")\n";

  return 0;
}
```

```
Point(double, double)
sPoint(double, double)
Point: (-1, -1)
Point2: (1, 1)
~sPoint(void)
~sPoint(void)

...Program finished with exit code 0
Press ENTER to exit console.
```

```
// object.h
#include <iostream>
struct sPoint
{
  sPoint(void) : x(0.0), y(0.0) {
    std::cout << "sPoint(void)\n"; return; }
  sPoint(double _x, double _y) : x(_x), y(-y) {
    std::cout << "sPoint(double, double)\n";
    return; }
  virtual ~sPoint(void) {
    std::cout << "~sPoint(void)\n"; return; }

  double x;
  double y;
};
```

C++ Objects
Operator Functions Example

```
#include <iostream>
#include "object.h"

int main(int argc, char *argv[])
{
  sPoint Point{-1.0, -1.0};
  sPoint Point2{x: 1.0, y: 1.0};

  std::cout << "Point: " << Point << "\n";
  std::cout << "Point2: " << Point2 << "\n";

  return 0;
}
```

```
Point: (-1, -1)
Point2: (1, 1)
~sPoint(void)
~sPoint(void)

...Program finished with exit code 0
Press ENTER to exit console.
```

```
// object.h
#include <iostream>
#include <ostream>
struct sPoint
{
  sPoint(void) : x(0.0), y(0.0) { return; }
  sPoint(double _x, double _y) : x(_x), y(_y)
{ return; }
```

```
 virtual ~sPoint(void) {
   std::cout << "~sPoint(void)\n";
   return; }
  double x;
  double y;
};

std::ostream & operator<<
    (std::ostream &File, const sPoint &Point)
{
  File << "(" << Point.x << ", "
    << Point.y << ")";

  return File;
}
```

C++ Objects
Operators That Can Be Overloaded

- These operators can be overloaded and made class specific:
  + - * / % ^ & | ~ ! = < > += -= *= /= %= ^= &= |= << >> >>= <<= == != <= >= && || ++ -- , ->* -> ( ) [ ]
- NOTE:
  - && and || do not short circuit when overloaded!
  - Operator precedence and priority cannot be changed.

C++ Objects
Operators That Can Be Overloaded

- Post-fix unary ++ and -- have a special syntax:
  <object> **`& operator ++(int)`**
  <object> **`& operator --(int)`**

C++ Objects throw Causes Destruction

```
#include <iostream>
#include "object.h"
int main(int argc, char *argv[])
{
  try {
    sPoint Point{-1.0, -1.0};
    sPoint Point2{x: 1.0, y: 1.0};
    throw "Fake error";
    std::cout << "Point: " << Point << “\n;
    std::cout << "Point2: " << Point2 << "\n";

  }
  catch (const char *e) {
    std::cout << "Error: " << e << std::endl;
  }
  return 0;
}
```

```
sPoint(void)
~sPoint(void)
Error: Fake error

...Program finished with exit code 0
Press ENTER to exit console.
```

```
// object.h
#include <iostream>
#include <ostream>
struct sPoint
{
  sPoint(void) : x(0.0), y(0.0) { return; }
  sPoint(double _x, double _y) : x(_x), y(_y)
{ return; }
```

```
 virtual ~sPoint(void) {
   std::cout << "~sPoint(void)\n";
   return; }
  double x;
  double y;
};

std::ostream & operator<<
    (std::ostream &File, const sPoint &Point)
{
  File << "(" << Point.x << ", "
    << Point.y << ")";

  return File;
}
```

C++ Objects
Copy Constructor Example

```
#include <iostream>
#include "object.h"

int main(int argc, char *argv[])
{
  sPoint Point{-1.0, -1.0};
  sPoint Point2{Point};

  std::cout << "Point: " << Point << "\n";
  std::cout << "Point2: " << Point << "\n";

  return 0;
}
```

```
Point: (-1, -1)
Point2: (-1, -1)
~sPoint(void)
~sPoint(void)

...Program finished with exit code 0
Press ENTER to exit console.
```

```
// object.h
#include <iostream>
#include <ostream>
struct sPoint
{
  sPoint(void) : x(0.0), y(0.0) { return; }
  sPoint(const sPoint &p) : x(p.x), y(p.y)
    { return; }
  sPoint(double _x, double _y) : x(_x), y(_y)
    { return; }
  virtual ~sPoint(void) {std::cout <<
    "~sPoint(void)\n"; return; }
```

```
  double x;
  double y;
};

std::ostream & operator<<
    (std::ostream &File, const sPoint &Point)
{
  File << "(" << Point.x << ", "
    << Point.y << ")";

  return File;
}
```

C++ Objects
`this`

- **`this`** is a pointer to the current object instance.
- Automatically created.
- **`*this`** is the instance.
- "**`->`**" operator is similar to "**`.`**" operator, but used with pointers.
- Syntax: **this->**<member>

C++ Objects
Assignment Operator Example

```
#include <iostream>
#include "object.h"

int main(int argc, char *argv[])
{
  sPoint Point{-1.0, -1.0};
  sPoint Point2;
  Point2 = Point;
  std::cout << "Point: " << Point << "\n";
  std::cout << "Point2: " << Point << "\n";

  return 0;
}
```

```
Point: (-1, -1)
Point2: (-1, -1)
~sPoint(void)
~sPoint(void)

...Program finished with exit code 0
Press ENTER to exit console.
```

```
// object.h
#include <iostream>
#include <ostream>
struct sPoint
{
  sPoint(void) : x(0.0), y(0.0) { return; }
  sPoint(const sPoint &p) : x(p.x), y(p.y)
   { return; }
  sPoint(double _x, double _y) : x(_x), y(_y)
    { return; }
  virtual ~sPoint(void) {std::cout <<
    "~sPoint(void)\n"; return; }
  sPoint &operator=(const sPoint &p)
    {x = p.x; y = p.y; return *this;}
  double x;
  double y;
};

std::ostream & operator<<
    (std::ostream &File, const sPoint &Point)
{
  File << "(" << Point.x << ", "
    << Point.y << ")";

  return File;
}
```

C++ Objects
Addition Operator Example

```
#include <iostream>
#include "object.h"

int main(int argc, char *argv[])
{
  sPoint Point{-1.0, -1.0};
  sPoint Point2{1.0, 1.0};

  std::cout << "Point + Point2: " <<
    Point + Point2 << std::endl;

  return 0;
}
```

```
Point + Point2: (0, 0)
~sPoint(void)
~sPoint(void)
~sPoint(void)

...Program finished with exit code 0
Press ENTER to exit console.
```

```
// object.h
#include <iostream>
#include <ostream>
struct sPoint
{
  sPoint(void) : x(0.0), y(0.0) { return; }
  sPoint(const sPoint &p) : x(p.x), y(p.y)
   { return; }
  sPoint(double _x, double _y) : x(_x), y(_y)
    { return; }
  virtual ~sPoint(void) {std::cout <<
    "~sPoint(void)\n"; return; }
  sPoint operator+(const sPoint &rhs)
  {sPoint Temp(this->x + rhs.x, this->y + rhs.y);
return Temp;}
```

```
  double x;
  double y;
};

std::ostream & operator<<
    (std::ostream &File, const sPoint &Point)
{
  File << "(" << Point.x << ", "
    << Point.y << ")";

  return File;
}
```

C++ Objects
+= Operator Example

```
#include <iostream>
#include "object.h"

int main(int argc, char *argv[])
{
  sPoint Point{-1.0, -1.0};
  sPoint Point2{1.0, 1.0};
  Point += Point2;
  std::cout << "Point += Point2: " <<
  Point << std::endl;

  return 0;
}
```

```
Point += Point2: (0, 0)
~sPoint(void)
~sPoint(void)

...Program finished with exit code 0
Press ENTER to exit console.
```

```
// object.h
#include <iostream>
#include <ostream>
struct sPoint {
  sPoint(void) : x(0.0), y(0.0) { return; }
  sPoint(const sPoint &p) : x(p.x), y(p.y)
    { return; }
  sPoint(double _x, double _y) : x(_x), y(_y)
    { return; }
  virtual ~sPoint(void) {std::cout <<
    "~sPoint(void)\n"; return; }
  sPoint &operator+=(const sPoint &rhs) {
    this->x += rhs.x; this->y += rhs.y;
    return *this;}
```

```
  double x;
  double y;
};

std::ostream & operator<<
    (std::ostream &File, const sPoint &Point)
{
  File << "(" << Point.x << ", "
    << Point.y << ")";

  return File;
}
```

C++ Objects
Member Function Example

```
#include <iostream>
#include "object.h"

int main(int argc, char *argv[])
{
  sPoint Point{0.0, 0.0};
  sPoint Point2{1.0, 1.0};

  std::cout << "Distance: " <<
    Point.Distance(Point2) << std::endl;
  return 0;
}
```

```
Distance: 1.41421

...Program finished with exit code 0
Press ENTER to exit console.
```

```
// object.h
#include <iostream>
#include <ostream>
#include <cmath>
struct sPoint
{
  sPoint(void) : x(0.0), y(0.0) { return; }
  sPoint(const sPoint &p) : x(p.x), y(p.y)
    { return; }
  sPoint(double _x, double _y) : x(_x), y(_y)
    { return; }
  virtual ~sPoint(void) {return; }
  double Distance(const sPoint &p) const
    {double dx = x - p.x; double dy = y - p.y;
    return std::sqrt(dx*dx + dy*dy);}
```

```
  double x;
  double y;
};

std::ostream & operator<<
    (std::ostream &File, const sPoint &Point)
{
  File << "(" << Point.x << ", "
    << Point.y << ")";

  return File;
}
```

C++ Objects
Default Methods

- C++ automatically provides these default methods:
  - Constructor: <object>**(void)**
  - Copy Constructor: <object>**(const** <object> **&o)**
  - Destructor: **~**<object>**(void)**
  - Assignment:
    <object> **&operator = (const** <object> **&**rhs**)**
- Default constructors can be deleted. e.g.,
  <object>**(void) = delete;**

C++ Objects
[ ] Operator Example

```
#include <iostream>
#include "object.h"

int main(int argc, char *argv[])
{
  sArray Array(11u);
  Array[1u] = 29u;
  std::cout << "Array[1u]: " << Array[1u]
     << std::endl;

  return 0;
}
```

```
Array[1u]: 29

...Program finished with exit code 0
Press ENTER to exit console.
```

```
// object.h
#include <iostream>
#include <cstdint>
#include <ostream>
#include <cstdlib>
struct sArray
{
  sArray(void) = delete;
  sArray(const uint32_t n) { Array = (double *)
    malloc(n * sizeof(double)); return; }
  virtual ~sArray(void) { free(Array); return; }
  double &operator[] (const uint32_t i)
    { return Array[i]; }

  double *Array;
};
```