C++ Programming
Lesson 6

TINPRO02-5B

HOGESCHOOL ROTTERDAM

PROGRAMMA

Recap

Stream I/O.

Operator overloading
Reference return
lvalue, &&

HOGESCHOOL ROTTERDAM

Recap Lesson 4

- Inheritance
- Multiple inheritance
- Abstract classes
- Polymorphism
- Virtual function overriding
- Virtual destructors

Cpp and h files: What goes where?

.h file:

- includes
- class definition (one or more classes in the same module)
- enums

.cpp file:

- includes
- class implementation

Cpp and h files

```
#include <iostream>   // Standard include path
#include "module.h"   // Current directory
```

- In this course only class declarations allowed in header
- No global #defines, constants, variables c.q instances
- No free functions

Example .h file

```
#include "window.h"

class GameObject{
	public:
		GameObject(Window *window, int color = COLOR_WHITE_BLACK,
			int colorSelected = COLOR_WHITE_BLACK);
		virtual ~GameObject() = 0;

		void setLocation(int x, int y);
		void setSize(int x, int y);

		void draw(bool selected = false);
		virtual void drawAtPos(int x, int y, bool selected = false) = 0;

		int getColor(bool selected = false);
	private:
		void drawBorder(int posX, int posY);
		virtual void drawAtPos(int x, int y, bool selected = false) = 0;

		//Variables
		int _posX;
		int _posY;
		int _sizeX;
		int _sizeY;
		Window *_window;
};
```

Example .cpp file (snippet)

```
#include "gameobject.h"

GameObject::GameObject(Window *window, int color, int colorSelected){
    _window = window;
    this->setLocation(0,0);
    _color = color;
    _colorSelected = colorSelected;
}

GameObject::~GameObject(){}

void GameObject::draw(bool selected){
    this->drawAtPos(_posX, _posY, selected);
}

void GameObject::setLocation(int x, int y){
    _posX = x;
    _posY = y;
}
```

Example

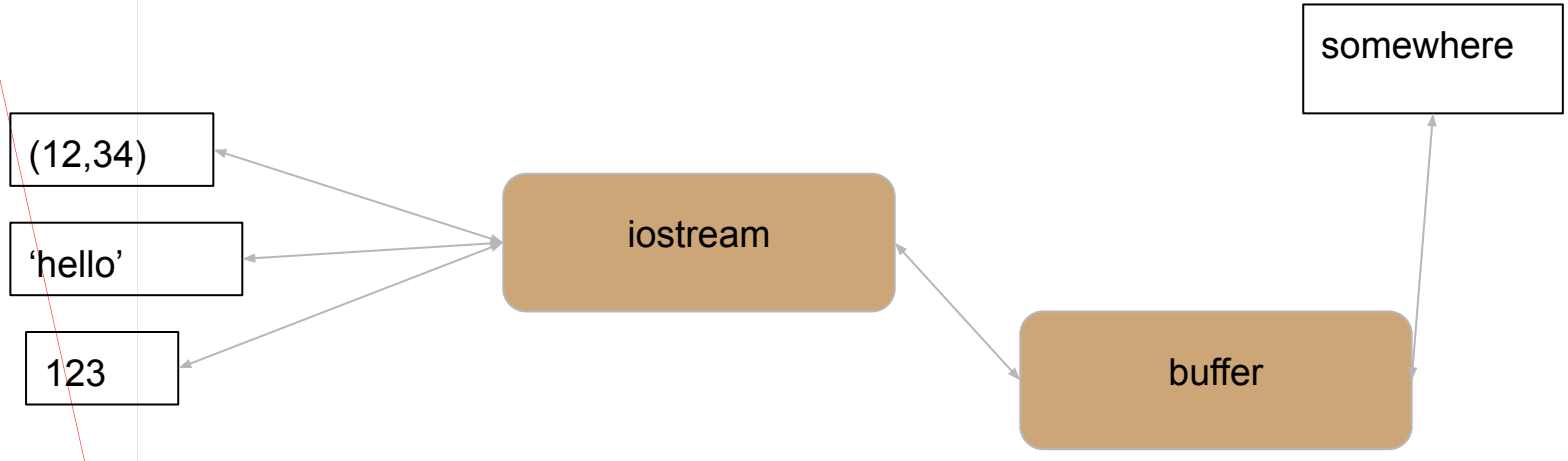
https://github.com/wouterbruggeman/spacewalk

Modules

- Large programs should be split into sections, or modules.
- C++ allows programs to be split into multiple files, compiled separately, and then combined (linked) to form a single program.
- A module is a collection of functions or classes that perform related functions.

Stream I/O: Console

- Already known and widely used in this course
- Library: <iostream>
- istream : input stream
- ostream : output stream



Stream I/O: File

- Same as for console:
  - **`ifstream`** read from a given file
  - **`ofstream`** write to a given file
  - **`fstream`** read and write to a given file
- To open a file with a stream object we use its member function open:
  - **`open (filename, mode);`**

Mode values	explanation
in	Input
out	Output
binary	Binary mode
ate	At end
app	Append
trunc	Truncate

HOGESCHOOL ROTTERDAM

Return value by reference

- A C++ function can return a reference in a similar way as it returns a pointer.
- When a function returns a reference, it returns an implicit pointer to its return value.
- A function can be used on the left side of an assignment statement.

Example

```
#include <iostream>
#include <ctime>

using namespace std;

double vals[] = {10.1, 12.6, 33.1, 24.1, 50.0};

double& setValues( int i ) {
   return vals[i];   // return a reference to the ith element
}

// main function to call above defined function.
int main () {
   setValues(1) = 20.23; // change 2nd element
   setValues(3) = 70.8;  // change 4th element
   return 0;
}
```

Be careful with reference returns!

- When returning a reference, be careful that the object being referred to does not go out of scope.
- It is not legal to return a reference to local var. But you can always return a reference on a static variable.

```
int& func() {
   int q;
   //! return q; // Compile time error
   static int x;
   return x;      // Safe, x lives outside this scope
}
```

L-values and R-values

- The term L-value is used for something that can appear on the left-hand side of an assignment operator.
- The term R-value is used for something that can appear on the right-handside of an assignment operator
- If you want the object return by a function to be an L-value, it must be returned by reference

Example

```
#include <iostream>
#include <ctime>

using namespace std;

double vals[] = {10.1, 12.6, 33.1, 24.1, 50.0};

double& setValues( int i ) {
   return vals[i];   // return a reference to the ith element
}

// main function to call above defined function.
int main () {
   setValues(1) = 20.23; // change 2nd element
   setValues(3) = 70.8;  // change 4th element
   return 0;
}
```

L-value

Operator Overloading

- It is possible to make operators to work for user defined classes.
- This means C++ has the ability to provide the operators with a special meaning for a data type, this ability is known as **operator overloading**.

For example, we can overload an operator '+' in a class like String so that we can concatenate two strings by just using +.

Or we can overload the operator '<<' and '>>' to input and output objects of user defined classes

Operator overloading example

```
#include <iostream>
using namespace std;

class Date{
   private:
      int mo, da, yr;
   public:
      Date(int m, int d, int y)
      {
         mo = m; da = d; yr = y;
      }
      friend ostream& operator<<(ostream& os, const Date& dt);
};

ostream& operator<<(ostream& os, const Date& dt){
    os << dt.mo << '/' << dt.da << '/' << dt.yr;
    return os;
}

int main(){
    Date dt(5, 6, 92);
    cout << dt;
}
```

The overloaded `<<` operator function must then be declared as a friend of class `Date` so it can access the private data within a `Date` object.

Operator overloading remarks

The overloaded operator returns a reference to the original `ostream` object, which means you can combine insertions:

```
…
{
    Date dt(5, 6, 92);
    Date dt2(16, 7, 71);
    cout << "The date is: "<<  dt << " Date
2 is: " << dt2 << endl;
}
```

Exercise

Following the example of the operator overloading for the class Date, write the operator overloading for the same class for the following operators:

>> (tip: use `istream` instead of `ostream`)

+ (adds day by day, month by month, year by year, add checks for max days per month and max 12 months per year)

- (similar to + but then with subtraction)

== (checks if two dates are the same)

overtref jezelf