EDAN30 Photorealistic Computer Graphics

C++

Michael Doggett
Department of Computer Science
Lund university

C++ and PCG

- Similarities with Java
  - which you should know by heart
- C++ used in all assignments

- You should really take Per Holm’s C++ course
  - Some of these slides are borrowed from Per

This seminar...

- ...covers only the features that you need in order to pass the assignments

- Thus, this seminar is **not** a full tour of C++
  - Again, take a look at the C++ course at the department of Computer Science

Comments

```
// a one line comment

/* a comment
   spanning
   several lines */

/** this works as well, but has no special
    meaning as in Java */
```

A class definition

```
class Point
{
  public:
      Point(float x, float y); // constructor
      float getX(void) const;  // accessor: const means
      float getY(void) const;  // does not change state
                               // of object
  protected:
      float mX;                // member attributes
      float mY;
};
```

This part of the code is in a header-file: point.h

The actual implementation is in point.cpp

Implementation of point

```
// point.cpp
Point::Point(float x,float y) // Point:: indicates
{                             // which class
  mX=x;
  mY=y;
}

float Point::getX(void) const // getY() in the same way
{
  return mX;
}
```

Declarations and definitions

Function declaration:

In header file (for example):

```
bool finished(int t);           // note semi-colon instead
                                // of function body
```

In cpp-file, function definition:

```
bool finished(int t)
{
  if(t>1) return true;
  else return false;
}
```

Brief on allocation

```
void func(void)   // function not belonging to class
{
  int a;
  a=sin(0.314);
  Point pl;
  Point *p=new Point(10.0, 20.0);   // a pointer
}
```

When `func()` is entered, `a` & `pl` are allocated on the stack, and when exited, `a` & `pl` are automatically deleted

`p` is allocated using new, which means that you need to delete it at some point: delete `p`;

There is **NO garbage collection** in C++.

Destructor

Used when a class allocates memory using new.

The destructor deletes what it has allocated

```
class Point
{
  public:
      Point();
      ~Point();           // destructor
};
```

In point.cpp:

```
Point::~Point()
{
  // delete memory here, for example:
  delete mNameOfPointString;   //if there was such a variable
}
```

Inheritance

```
class Point
{
  public:
      virtual void update(void);
};

class TimePoint : public Point   // inherit from Point
{
  public:
      void update(void);       // overloads Point::update
};
```

Namespaces

Similar to packages in Java.

In header-file, asr.h:

```
namespace asr
{
  class Point {...};
  void func(bool b);
}
```

In cpp.file:

```
#include ”asr.h”
namespace asr
{
  float Point::getX(void) const { return mX; }
}
```

Using namespaces

```
#include ”asr.h”

void test(void)
{
  using asr::Point;
  Point p;
}

// can ”import” everything from a namespace by
using namespace asr;
```

Primitive data types

`char`	`8 bit character`
`int`	`32 bit integer`
`short`	`16 bit integer`
`long`	`32 bit integer (usually)`
`float`	`32 bit single precision floating point`
`double`	`64 bit double precision floating point`
`bool`	`true or false`

The number of bits above are common values, but these may not hold for every architecture

Integers and char may be ”signed” or ”unsigned”

For example:

```
unsigned short a;   // 16 bit no sign

const float cPi=3.141592;        // constants
```

Operators

Almost identical to Java

`Arithmetic:`	**`*, /, %, +, -`**
`Relational:`	**`<, <=, ==, >=, >, !=`**
`Logical:`	**`!,  &&, ||`**
`Bitwise:`	**`& (bitwise AND), ^ (bitwise XOR)`**
	`| (bitwise OR), ~ (complement)`
	`<< (left shift), >> (right shift)`
`Special:`	**`?`**
	`+=, -=, *=, /=, ...`
	`x++; ++x; --x; x--;`

Control structures

```
// if, switch, while, do while, for : as in Java

// Example:
for(int q=0; q<N; q++)
{
   ...
}
```

Parameter passing

Default as in Java: parameter is copied

Then we have pointers and references as well

```
int func(Point &pr, Point *pp) // & is ref, * is pointer
{
   pr.set(10,10);
   pp->set(20,20);
}

....
Point p1; Point *p2=new Point(1,2);
func(p1,p2);
// p1 will hold 10,10, p2 will hold 20,20
// references are *not* copied (can be faster for large params)
```

Reference and pointer parameters

```
void addRef10(Point &p)
{
  p.setX( p.getX()+10 );                // note .
}

void addPtr10(Point *p)
{
  p->setX( p->getX()+10 );              // note ->
}

///
void test(void)
{
  Point p(1,1);  Point *pp= new Point(1,1);
  addRef10(p);
  addPtr10(pp)
}
```

Functions and arrays

An array:

```
int a[10];  // no a.length as in Java
```

Array is passed as pointer to first element:

```
void func(int b[], int sizeOfArray) {...}
```

or

```
void func(int *b, int sizeOfArray) {...}
```

Simple output...

```
// For example, might be good for debugging sometimes
#include <streamio.h>

std::cout << "raytracing..." << std::endl;

// prints ”raytracing...” to standard output
```

Want to know more?

- There is an enormous amount of information on the internet.
    - From web pages (cplusplus.com) to forums (stackoverflow.com)
- Have a look at the slides from Per Holm's C++ course.
    - There are pointers to lots of material for self-studies on course website.
- Plenty of good textbooks as well.