CMSC341 Data Structures

Lecture 1

Sept 6, 2000

Administrivia

Description
- prereqs
- grading
- responsibilities

Syllabus

Web page
- www.cs.umbc.edu/courses/undergraduate/341/fall00

Projects

Projects

- grading
- due dates
- honesty
- submittal: cs341-04
- project guidelines
- project 0 -- optional
- project 1 -- Due Sept 25

Preview Sessions:

- Wed/Thurs, Sept 6/7 8:30-9:30, SS114: C++/env
- Wed/Thurs, Sept 13/14 8:30-9:30, SS114: Proj1

C++ Class

Class organizes data and operations on that data

encapsulation

abstraction

Definition in header file, .H (.h in MAW)

Implementation in source file, .C (.cpp in MAW)

#include the header file

IntCell.H

```
#ifndef _IntCell_H_
#define _IntCell_H_
class IntCell {
  public:
      explicit IntCell (int initialValue = 0);
      IntCell (const IntCell &ic);
      ~IntCell();
      const IntCell &
            operator=(const IntCell &rhs);
      int read() const;       // accessor
      void write(int x);      // mutator
  private:
      int _storedValue;
  };
#endif
```

IntCell.C

```
#include “IntCell.H”

IntCell::IntCell(int initialValue) :
  _storedValue(initialValue)
  {   }
IntCell::IntCell (const IntCell &ic)
  {
  write(ic.read());
  }
IntCell:: ~IntCell()
  {   }
```

IntCell.C (cont)

```
const IntCell &IntCell::operator=(const IntCell &rhs)
  {
  if (this != &rhs)
      write (rhs.read())
  return *this;
  }
int IntCell::read() const
  {
  return _storedValue;
  }
void IntCell::write(int x)
  {
  _storedValue = x;
  }
```

TestIntCell.C

```
#include “IntCell.H”
#include <iostream.h>      // for cout
#include<stdlib.h>         // for EXIT_SUCCESS
int main()
  {
  IntCell m; // Or IntCell m(0);, but not IntCell m();
  IntCell n;
  n = m;
  m.write(5);
  cout << “Cell m contents: “ << m.read() << endl;
  cout << “Cell n contents: “ << n.read() << endl;
  return (EXIT_SUCCESS);
  }
```