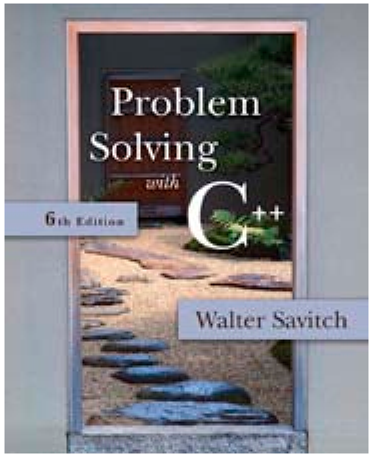
APS105: Lecture 25

Wael Aboelsaadat

wael@cs.toronto.edu

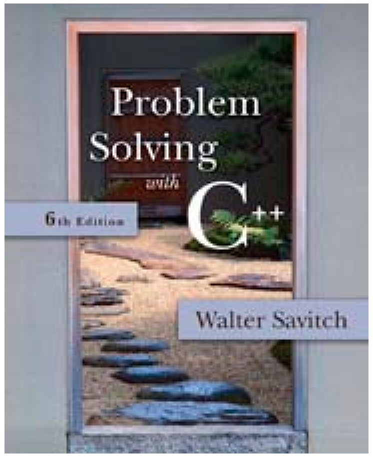
http://ccnet3.utoronto.ca/20079/aps105h1f/

Acknowledgement: These slides are a modified version of the text book slides as supplied by Addison Wesley



Chapter 9

Pointers and Dynamic Arrays



Type Definitions

- A name can be assigned to a type definition, then used to declare variables
- The keyword typedef is used to define new type names
  - Syntax:

    typedef Known_Type_Definition New_Type_Name;

    - Known_Type_Definition can be any type

Defining Pointer Types

- To avoid mistakes using pointers, define a pointer type name
  - Example: typedef int* IntPtr;

    Defines a new type, IntPtr, for pointer variables containing pointers to int variables
  - IntPtr p;
    is equivalent to
    int *p;

Multiple Declarations Again

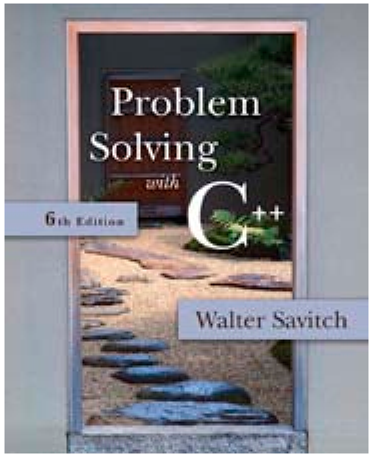
- Using our new pointer type defined as
  typedef int* IntPtr;
  - Prevent this error in pointer declaration:
    int *P1, P2; // Only P1 is a pointer variable
- with
  IntPtr P1, P2; // P1 and P2 are pointer
  // variables

Section 9.1 Conclusion

- Can you
  - Declare a pointer variable?
  - Assign a value to a pointer variable?
  - Use the new operator to create a new variable in the freestore?
  - Write a definition for a type called NumberPtr to be a type for pointers to dynamic variables of type int?
  - Use the NumberPtr type to declare a pointer variable called my_point?

Chapter 14

Recursion



```
#include <iostream>

using namespace std;


void exec( int nVar )
{
   int iIndex;

    cout << "inside exec 1..  " << nVar << endl;

    nVar++;

    if( nVar == 5 )    // base condition
      return;
    else
        exec( nVar  ); // causing the recursion

    cout << "---------------------------" << endl;
    cout << "inside exec 2.. " << nVar << endl;
}

int main( )
{

   exec(  0 );

   return 0;

}
```