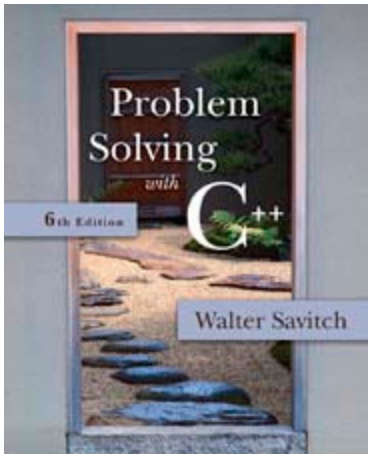
APS105: Lecture 29

Wael Aboelsaadat

wael@cs.toronto.edu

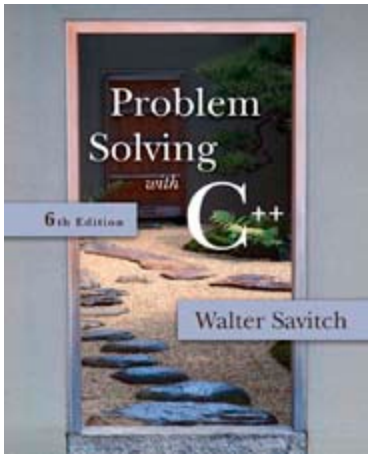
http://ccnet3.utoronto.ca/20079/aps105h1f/

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



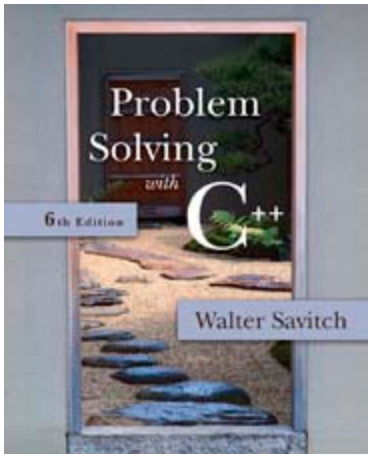
Chapter 14

Recursion



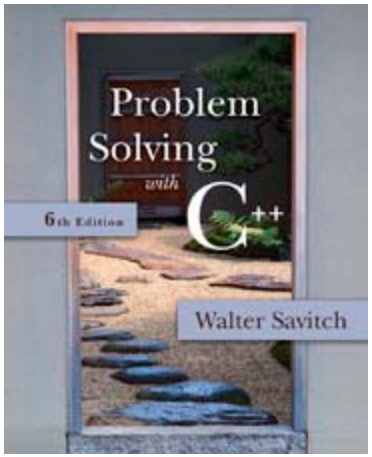
```
void MergeSort(int ar[], int left, int right, int pivot)
{
    if(left == right)
        return;
    else
    {
        MergeSort(ar, left, pivot, (left + pivot) / 2);
        MergeSort(ar, pivot + 1, right, (pivot + right + 1) / 2);
    }

    int LeftIndex  = left,
        PivotIndex = pivot + 1;
    while(PivotIndex != right + 1 && LeftIndex != PivotIndex) //continue until either list runs out
    {
        if(ar[PivotIndex] <= ar[LeftIndex])
        {
            int i;
            int iSrc = PivotIndex;
            int iDest= LeftIndex;
            int StoreSrc = ar[iSrc];
            for(i = iSrc; i > iDest; i --)
                ar[i] = ar[i - 1];     // Shifts numbers from iDest to iSrc one step forward
            ar[iDest] = StoreSrc;      // Puts final element in the right place
            PivotIndex++;
            LeftIndex++;
        }
        else
            LeftIndex++; // Skip to the next element
    }
}
```

Chapter 10



10.1

Structures



Structures

- A structure can be viewed as an object
  - Contains no member functions
    (The structures used here have no member functions)

  - Contains multiple values of possibly different types
    - The multiple values are logically related as a single item
    - Example: An employee record has the following values:
      a name
      a SIN
      a Salary
      an address

The CD Definition

- The Certificate of Deposit structure can be defined as

```
struct Employee
{
        string strName;
        string strAddress;
        int    Salary;
        int    SIN;
};
```

Remember this semicolon!

- Keyword struct begins a structure definition
- Employee is the structure tag or the structure's type
- Member names are identifiers declared in the braces

Using the Structure

- Structure definition is generally placed outside any function definition
  - This makes the structure type available to all code that follows the structure definition
- To declare two variables of type Employee:

```
Employee  employee1, employee2;
```

- employee1 and employee2 contain distinct member variables strAddress, Salary, etc..

The Structure Value

- The Structure Value
  - Consists of the values of the member variables

- The value of an object of type Employee
  - Consists of the values of the member variables

strName
strAddress
Salary;
SIN;

Specifying Member Variables

- Member variables are specific to the structure variable in which they are declared
  - Syntax to specify a member variable:
    Structure_Variable_Name . Member_Variable_Name
  - Given the declaration:
    Employee employee1, employee2;
    - Use the dot operator to specify a member variable
      employee1.strName
      employee1.strAddress
      employee1.SIN

Using Member Variables

- Check employee101.c

Duplicate Names

- Member variable names duplicated between structure types are not a problem.

```
struct FertilizerStock
{
   double quantity;
   double nitrogen_content;
};

FertilizerStock  super_grow;
```

```
struct CropYield
{
   int quantity;
   double size;
};

CropYield  apples;
```

- super_grow.quantity and apples.quantity are different variables stored in different locations

Structures as Arguments

- Structures can be arguments in function calls
  - The formal parameter can be call-by-value
  - The formal parameter can be call-by-reference
- Example:
  void get_data(Employee& employee);
  - Uses the structure type Employee we saw earlier as the type for a call-by-reference parameter

Structures as Return Types

- Structures can be the type of a value returned by a function
- Example:

```
Employee initialize (  )
{
    Employee employee;
    /// some initialization code
  return employee;
}
```

Using Function initialize

- initialize builds a complete structure value in employee, which is returned by the function
- We can use initialize to give a variable of type Employee a value in this way:

```
Employee  employee;
employee = initialize( );
```

Assignment and Structures

- The assignment operator can be used to assign values to structure types
- Using the Employee structure again:

```
Employee employee1, employee2;
employee1.strName  = “john”;
employee1.SIN        =  123123;
employee2 = employee1;
```

  - Assigns all member variables in employee2 the corresponding values in employee1

Hierarchical Structures

- Structures can contain member variables that are also structures

```
struct Date
{
   int month;
   int day;
   int year;
};
```

```
struct PersonInfo
{
    double height;
    int weight;
    Date birthday;
};
```

- struct PersonInfo contains a Date structure

Using PersonInfo

- A variable of type PersonInfo is declared by
  PersonInfo person1;
- To display the birth year of person1, first access the birthday member of person1

  cout << person1.birthday…

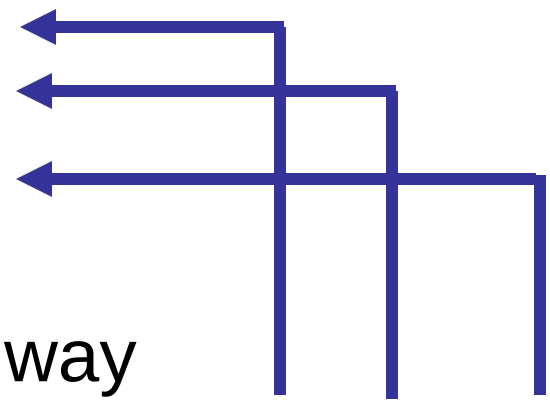
- But we want the year, so we now specify the year member of the birthday member

  cout << person1.birthday.year;

Initializing Structures

- A structure can be initialized when declared
- Example:

```
struct Date
{
        int month;
        int day;
        int year;
};
```



  - Can be initialized in this way

```
Date  due_date = {12, 31, 2004};
```