UAH CPE 112

File Input and Output

- A file is a named area in secondary storage that holds a collection of information (on disk, for example)
- Using files requires us to
  - Request the preprocessor to include the header file **fstream**
  - Use declaration statements to declare the file streams we are going to use
  - Prepare each file for reading or writing by using a function named open
  - Specify the name of the file stream in each input or output statement

UAH CPE 112

Using Files

- All of the **cin** and **cout** operations work for files.
- Declaring File Streams

```
ifstream inFile;
ofstream outFile;
```

(With these data types, you cannot read from and write to the same file)

- Opening Files

```
inFile.open("cone.dat");
outFile.open("results.dat");
```

(Open associates a stream variable used in your program with a physical file on disk.)

- Specifying File Streams in Input/Output Statements

```
inData >> htInInches >> diamInInches >> redPrice
       >> bluePrice >> greenPrice;
outData << "The painting cost for" << endl;
```

UAH CPE 112

An Example Program Using Files

```
//*************************************************************
// ConePaint program
// This program computes the cost of painting traffic cones in
// each of three different colors, given the height and diameter
// of a cone in inches, and the cost per square foot of each of
// the paints, all of which are input from a file
//*************************************************************
#include <iostream>
#include <iomanip>    // For setw() and setprecision()
#include <cmath>      // For sqrt()
#include <fstream>    // For file I/O

using namespace std;

const float INCHES_PER_FT = 12.0;   // Inches in 1 foot
const float PI = 3.14159265;        // Ratio of circumference
                                    //   to diameter
```

UAH CPE 112

An Example Program Using Files

```
int main()
{
    float    htInInches;    // Height of the cone in inches
    float    diamInInches;  // Diameter of base of cone in inches
    float    redPrice;      // Price per square foot of red paint
    float    bluePrice;     // Price per square foot of blue paint
    float    greenPrice;    // Price per square foot of green paint
    float    heightInFt;    // Height of the cone in feet
    float    diamInFt;      // Diameter of the cone in feet
    float    radius;        // Radius of the cone in feet
    float    surfaceArea;   // Surface area in square feet
    float    redCost;       // Cost to paint a cone red
    float    blueCost;      // Cost to paint a cone blue
    float    greenCost;     // Cost to paint a cone green
    ifstream inData;        // Holds cone size and paint prices
    ofstream outData;       // Holds paint costs
```

UAH CPE 112

An Example Program Using Files

```
   outData << fixed << showpoint;          // Set up floating-pt.
                                           //   output format
    // Open the files

    inData.open("cone.dat");
    outData.open("results.dat");

    // Get data

    inData >> htInInches >> diamInInches >> redPrice
           >> bluePrice >> greenPrice;

    // Convert dimensions to feet

    heightInFt = htInInches / INCHES_PER_FT;
    diamInFt = diamInInches / INCHES_PER_FT;
    radius = diamInFt / 2.0;
```

UAH CPE 112

An Example Program Using Files

```
    // Compute surface area of the cone

    surfaceArea = PI * radius *
                  sqrt(radius*radius + heightInFt*heightInFt);

    // Compute cost for each color

    redCost = surfaceArea * redPrice;
    blueCost = surfaceArea * bluePrice;
    greenCost = surfaceArea * greenPrice;
```

An Example Program Using Files

```
   // Output results

   outData << setprecision(3);
   outData << "The surface area is " << surfaceArea << " sq. ft."
           << endl;
   outData << "The painting cost for" << endl;
   outData << "   red is" << setw(8) << redCost << " dollars"
           << endl;
   outData << "   blue is" << setw(7) << blueCost << " dollars"
           << endl;
   outData << "   green is" << setw(6) << greenCost << " dollars"
           << endl;
   return 0;
}
```

Run-Time Input of File Names

- **inFile.open(“datafile.dat”)** restricts us, we can’t run that program on another file, we’d like to make the file name an input to the program
- To do so, we need code like the following:

```
ifstream inFile;
string fileName;

cout << “Enter the input file name: ”;
cin >> filename;
inFile.open(fileName.c_str());
```

Input Failure (How Things Can Go Very Wrong)

- If an input operation fails because of invalid data, the **cin** stream enters the silent but deadly fail state. (No error)
- Example:

```
int i = 10, j = 20, k = 30;
cin >> i >> j >> k;
cout << i: ” << i << “  j: ” << j
     << “  k: ” << k;
```

Input Data 1234.56 7 89 produces this output:

```
i: 1234  j: 20  k: 30
```

Software Design Methodologies

- Object-Oriented Design
  - A technique for developing software in which the solution is expressed in terms of self-contained entities composed of data and operations on that data (objects).
- Functional Decomposition
  - A technique for developing software in which the problem is divided into more easily handled subproblems, the solution of which creates a solution to the overall problem.

Software Design Methodologies

- Object-Oriented Design
  - A technique for developing software in which the solution is expressed in terms of self-contained entities composed of data and operations on that data (objects).
- Functional Decomposition
  - A technique for developing software in which the problem is divided into more easily handled subproblems, the solution of which creates a solution to the overall problem.

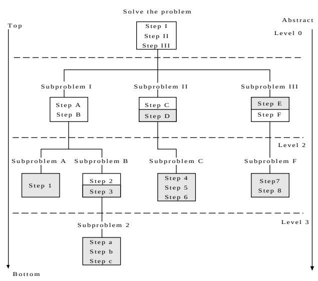
What are Objects?

- An object is a collection of data together with associated operations.
- In object-oriented programming languages, a class is a programmer-defined data type from which objects are created.
- **iostream** defines the classes **istream** and **ostream** and declares **cin** and **cout**
  - **istream cin;**
  - **istream cout;**
- **string** is also a class.

Functional Decomposition

- Functional decomposition is also known as structured design, top-down design, stepwise refinement and modular programming.
- By subdividing the problem, you create a hierarchical structure called a tree structure.
- Each level of the tree is more concrete than the level above it.

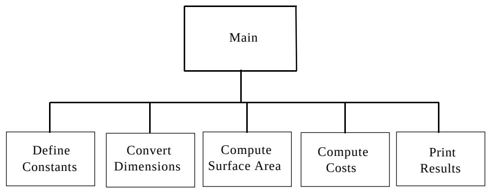
Hierarchical Structure Tree



Modules

- A module begins life as an abstract step in the next-higher level of the solution tree. It is completed when it solves a subproblem.
- Writing Cohesive Modules
  - Think
  - Begin writing major steps
  - If a step is simple enough, stop
  - If you have to break a step up, it's still abstract
  - If you have something hopelessly complicated, go back up a level

Functional Decomposition Example



Main Module

Define constants

Convert dimensions to feet

Set radius = diamInFt/2

Compute surface area

Compute costs

Print results

Define Constants

```
HT_IN_INCHES = 30.0
DIAM_IN_INCHES = 8.0
INCHES_PER_FT = 12.0
RED_PRICE = 0.10
BLUE_PRICE = 0.15
GREEN_PRICE = 0.18
PI = 3.14159265
```

Convert dimensions to Feet

Set heightInFt = HT_IN_INCHES/INCHES_PER_FT

set diamInFt = DIAM_IN_INCHES/INCHES_PER_FT

Compute Surface Area

Set surfaceArea = pi x radius xsqrt(radius^2 + heightInFt^2)

Print Results

Print surfaceArea

Print redCost

Print blueCost

Print green Cost

Problem-Solving Case Study

- Problem: For your art class, you need to know how much wood to buy for a frame, how large a piece of canvas to purchase, and the cost of the materials for a given for a given painting size.
- Inputs: length and width of the painting (**float**), the cost per inch of wood (**float**), and the cost per square foot of canvas (**float**).
- Outputs: Prompting messages, the input data, length of wood to buy (**float**), the dimensions of the canvas (**float**), the cost of the wood (**float**), the cost of the canvas (**float**), and the total cost of the materials (**float**).

Problem-Solving Case Study (continued)

- Discussion: The length of the wood is twice the sum of the length and the width of the painting. The cost of the wood is simply its length times its cost per inch. For the canvas, we must add 5 inches to the length and width to allow for wrapping the canvas around the frame.The area is length times width, converted from square inches to square feet.
- Assumptions: The input values are positive (checking for erroneous data is not done).

Main Module

Get length and width

Get wood and canvas costs

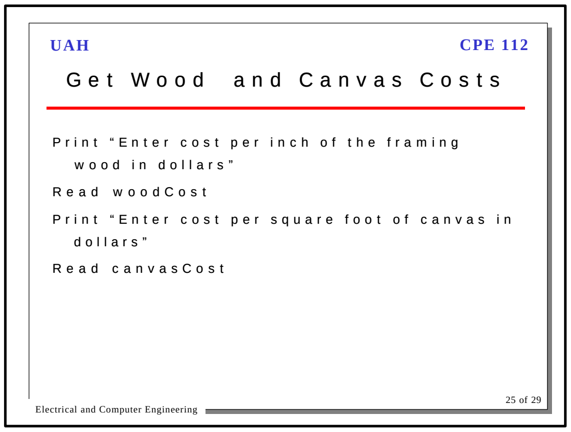
Compute dimensions and costs

Print dimensions and costs

Get Length and Width

Print “Enter length and width of painting”

Read length, width

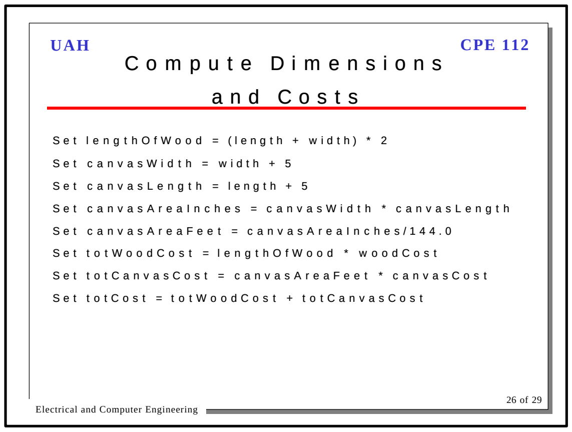


UAH CPE 112

Get Wood and Canvas Costs

```
Print "Enter cost per inch of the framing
    wood in dollars"
Read woodCost
Print "Enter cost per square foot of canvas in
    dollars"
Read canvasCost
```

Electrical and Computer Engineering 25 of 29

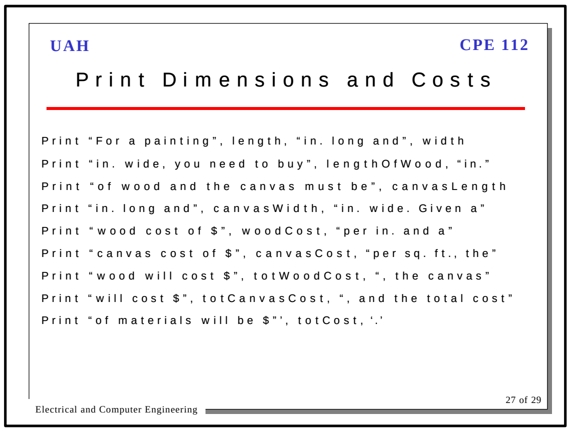


UAH CPE 112

Compute Dimensions and Costs

```
Set lengthOfWood = (length + width) * 2
Set canvasWidth = width + 5
Set canvasLength = length + 5
Set canvasAreaInches = canvasWidth * canvasLength
Set canvasAreaFeet = canvasAreaInches/144.0
Set totWoodCost = lengthOfWood * woodCost
Set totCanvasCost = canvasAreaFeet * canvasCost
Set totCost = totWoodCost + totCanvasCost
```

Electrical and Computer Engineering 26 of 29

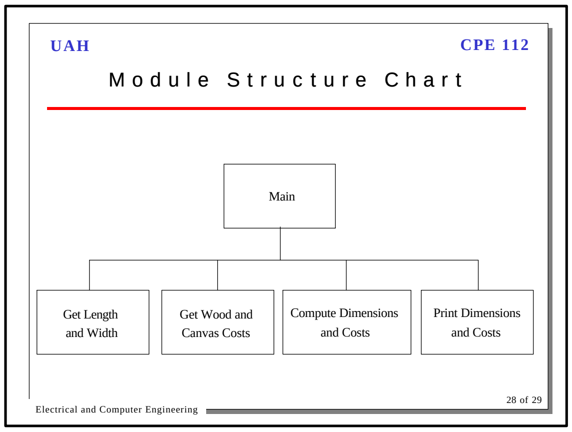


UAH CPE 112

Print Dimensions and Costs

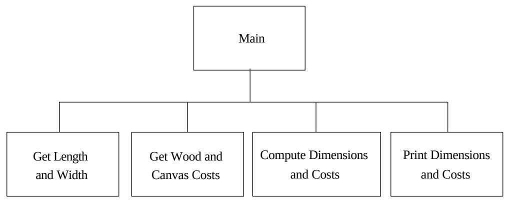
```
Print "For a painting", length, "in. long and", width
Print "in. wide, you need to buy", lengthOfWood, "in."
Print "of wood and the canvas must be", canvasLength
Print "in. long and", canvasWidth, "in. wide. Given a"
Print "wood cost of $", woodCost, "per in. and a"
Print "canvas cost of $", canvasCost, "per sq. ft., the"
Print "wood will cost $", totWoodCost, ", the canvas"
Print "will cost $", totCanvasCost, ", and the total cost"
Print "of materials will be $"', totCost, '.'
```

Electrical and Computer Engineering 27 of 29

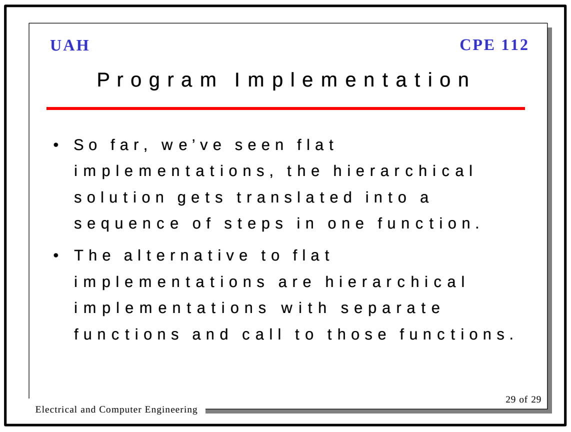


UAH CPE 112

Module Structure Chart


Main
Get Length and Width
Get Wood and Canvas Costs
Compute Dimensions and Costs
Print Dimensions and Costs

Electrical and Computer Engineering 28 of 29



UAH CPE 112

Program Implementation

- So far, we've seen flat implementations, the hierarchical solution gets translated into a sequence of steps in one function.
- The alternative to flat implementations are hierarchical implementations with separate functions and call to those functions.

Electrical and Computer Engineering 29 of 29