CIS 22A – Lecture 3

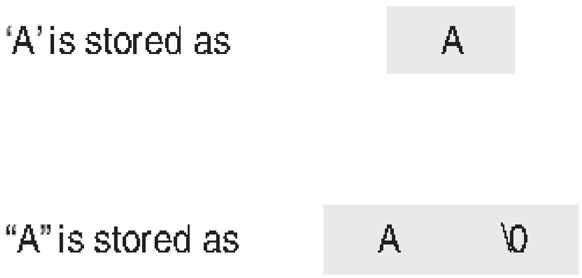
Manish Goel

Variable Scope

- Scope is very important in C++
- The scope of a variable: from when variable is initialized to when it is destroyed
- A variable cannot be used before it is defined
- The scope of a variable: the part of the program in which the variable can be accessed
- Trying to access a variable out of scope crashes the program – NULL pointer error

Gaddis Ch. 2 – Char vs String

Figure 2-6

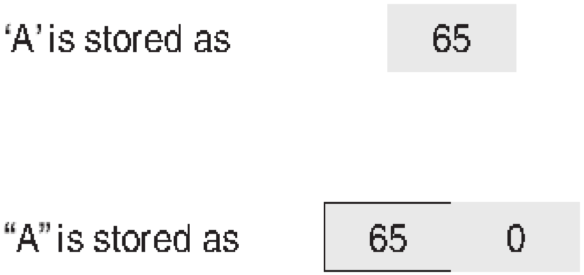


‘A’ is stored as A

“A” is stored as A \0

As you can see, ‘A’ is a 1-byte element and “A” is a 2-byte element. Since characters are really stored as ASCII codes, Figure 2-7 shows what is actually being stored in memory.

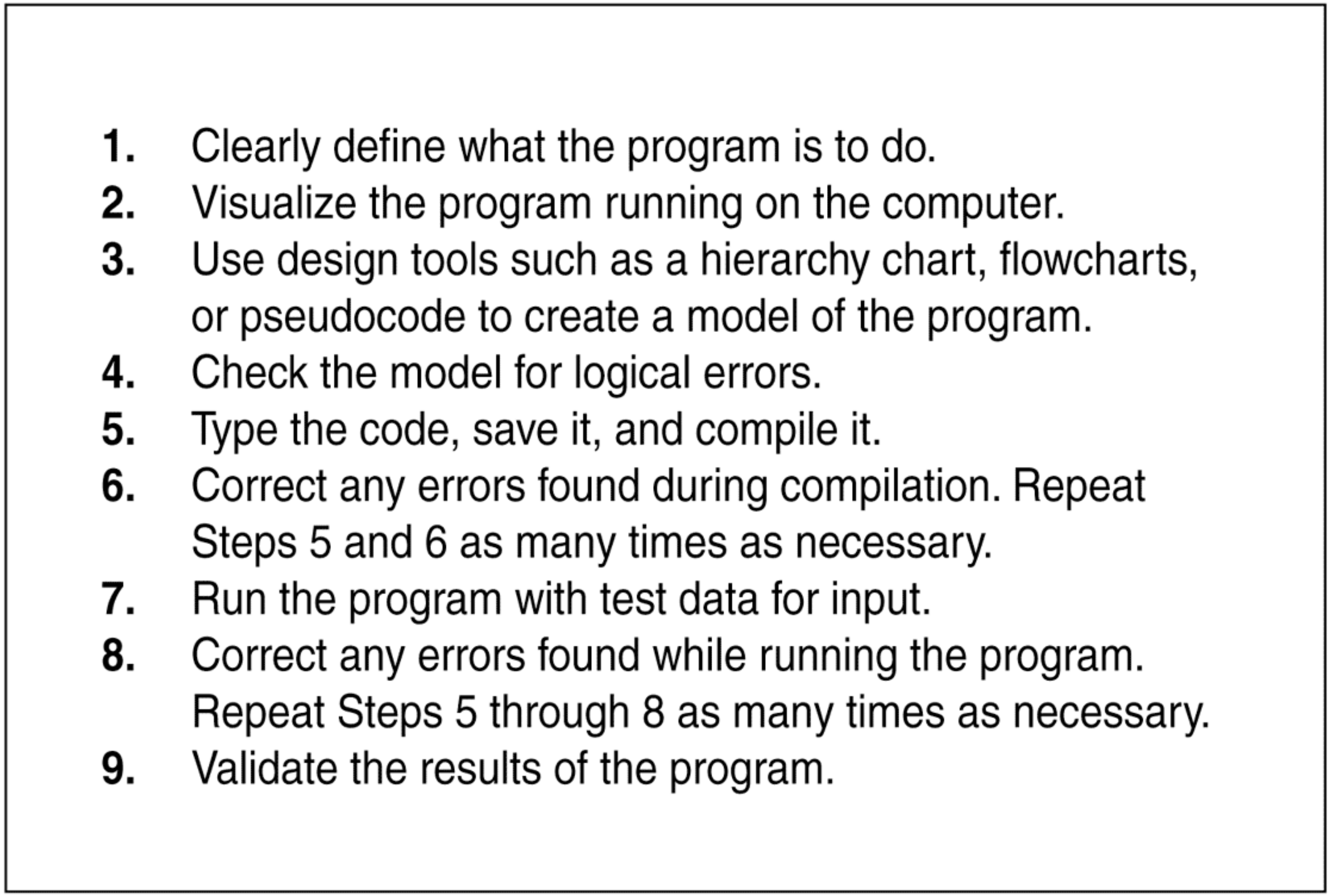
Figure 2-7



‘A’ is stored as 65

“A” is stored as 65 0

The Programming Process



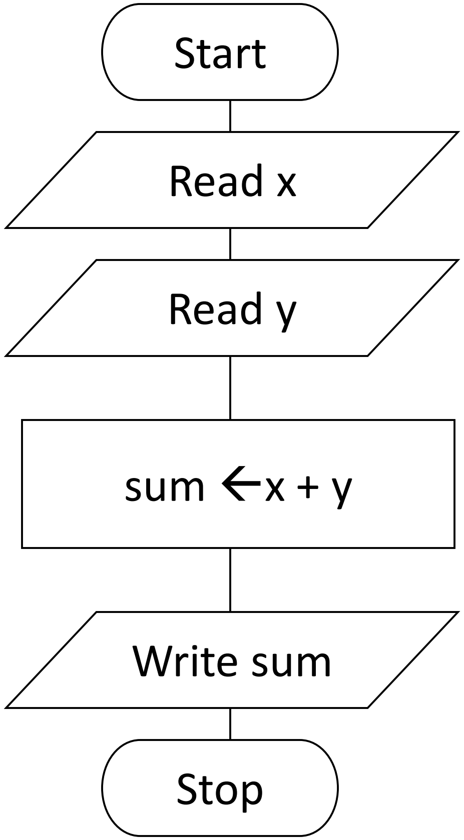
1. Clearly define what the program is to do.
2. Visualize the program running on the computer.
3. Use design tools such as a hierarchy chart, flowcharts, or pseudocode to create a model of the program.
4. Check the model for logical errors.
5. Type the code, save it, and compile it.
6. Correct any errors found during compilation. Repeat Steps 5 and 6 as many times as necessary.
7. Run the program with test data for input.
8. Correct any errors found while running the program. Repeat Steps 5 through 8 as many times as necessary.
9. Validate the results of the program.

PROBLEM

Design a solution for a program that adds two numbers.

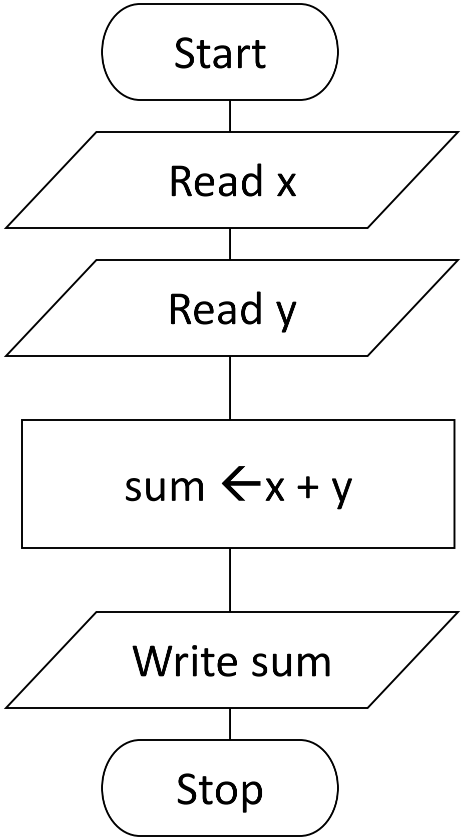
1. Get the first number
2. Get the second number
3. Add the two numbers
4. Display the result

1. Get the first number
2. Get the second number
3. Add the two numbers
4. Display the result



```
1.   Get the first number
2.   Get the second number
3.   Add the two numbers
4.   Display the result
```

pseudocode



Start

Read x

Read y

sum ← x + y

Write sum

Stop

flow chart

Program Development:

1. Understand the problem

2. Develop a solution using structure charts and either flow-charts or pseudo-code

3. Write the program

4. Test the program

Program Development:

1. Understand the problem
    - A. What do I know?
    - B. What do I have to do?
    - C. How do I get from (A) to (B)?

2. Develop a solution using structure charts and either flow-charts or pseudo-code

3. Write the program

4. Test the program

PROBLEM

Design a solution for a program that calculates the area and the perimeter of a circle.

PROBLEM

Design a solution for a program that calculates the area and the perimeter of a circle.

1. Understand the problem
    - A. What do I know?
    - B. What do I have to do?
    - C. How do I get from (A) to (B)?

PROBLEM

Design a solution for a program that calculates the area and the perimeter of a circle.

1. Understand the problem

   A. What do I know?

      – the radius, r, and π

   B. What do I have to do?

      – calculate the area and the circumference

   C. How do I get from (A) to (B)?

      area $= \pi r^2$

      circ $= 2 \pi r$

Program Development:

1. Understand the problem
2. *Develop a solution using structure charts and either flow-charts or pseudo-code*
3. Write the program
4. Test the program

1. Understand the problem

   A. What do I know?

   – the radius, r, and π

   B. What do I have to do?

   – calculate the area and the circumference

   C. How do I get from (A) to (B)?

   area $= \pi r^2$

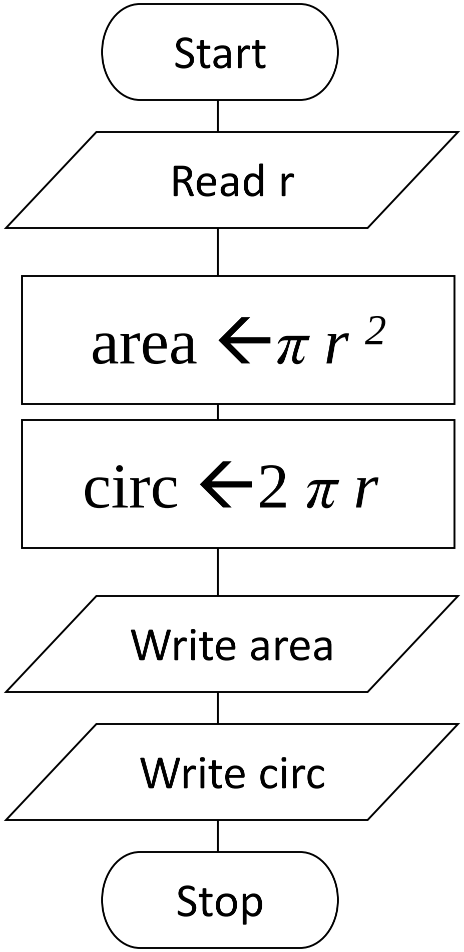
   circ $= 2 \pi r$

```
1. Get the radius, r
2. Calculate the area:
           area = $\pi r^2$
3. Calculate the circumference:
           circ = $2 \pi r$
4. Display area
5. Display circ
```

pseudocode

```
1. Get the radius, r
2. area = $\pi r^2$
3. circ = $2 \pi r$
4. Display area
5. Display circ
```

pseudocode



Start

Read r

area $\leftarrow \pi r^2$

circ $\leftarrow 2 \pi r$

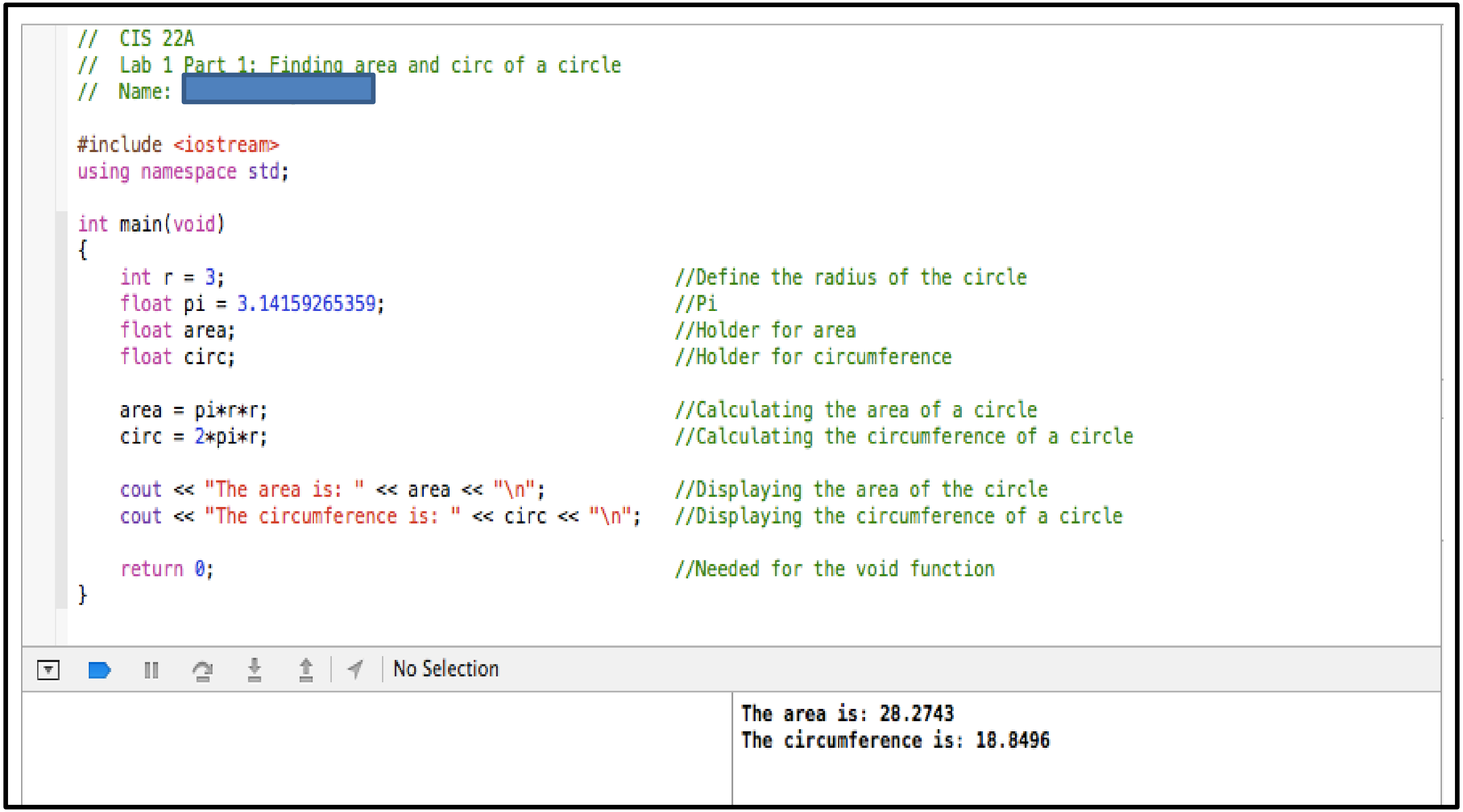
Write area

Write circ

Stop

flow chart

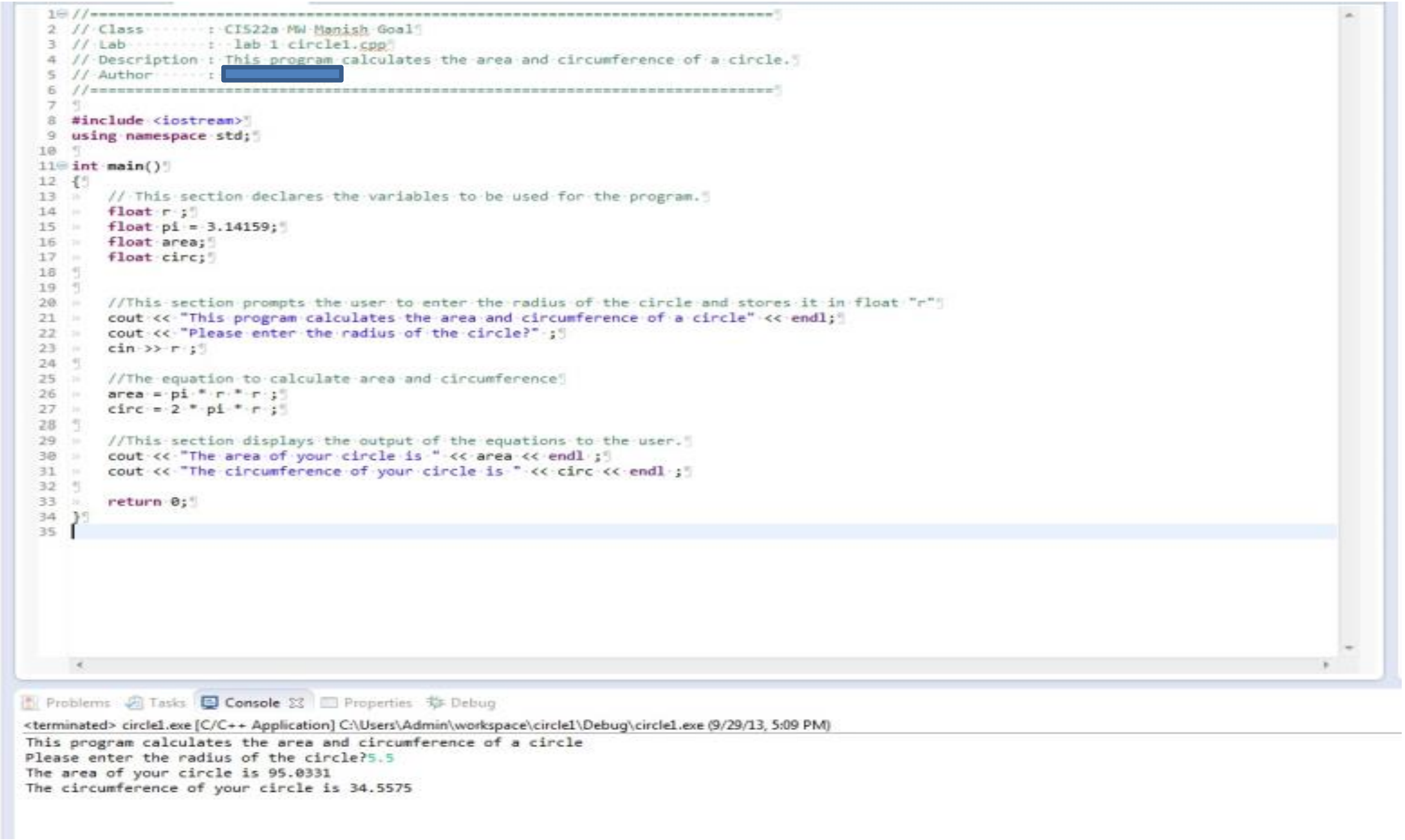
Lab 1 – Good Programming Style (1)



```cpp
//  CIS 22A
//  Lab 1 Part 1: Finding area and circ of a circle
//  Name: [redacted]

#include <iostream>
using namespace std;

int main(void)
{
    int r = 3;                                         //Define the radius of the circle
    float pi = 3.14159265359;                          //Pi
    float area;                                        //Holder for area
    float circ;                                        //Holder for circumference

    area = pi*r*r;                                     //Calculating the area of a circle
    circ = 2*pi*r;                                     //Calculating the circumference of a circle

    cout << "The area is: " << area << "\n";           //Displaying the area of the circle
    cout << "The circumference is: " << circ << "\n";  //Displaying the circumference of a circle

    return 0;                                          //Needed for the void function
}
```

```
The area is: 28.2743
The circumference is: 18.8496
```

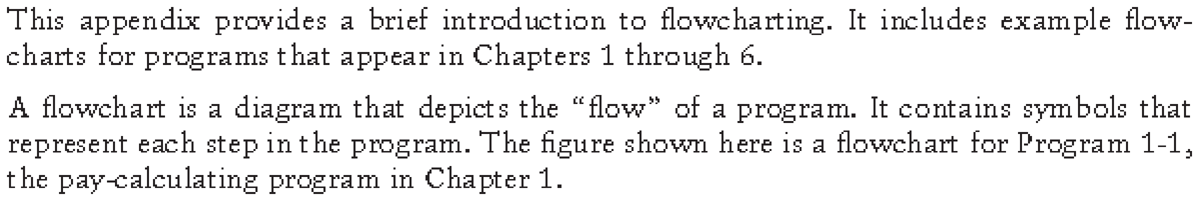
Lab 1 – Good Programming Style (2)



```
//==============================================================================
// Class       : CIS22a MW Manish Goal
// Lab         :  lab 1 circle1.cpp
// Description : This program calculates the area and circumference of a circle.
// Author      : [redacted]
//==============================================================================

#include <iostream>
using namespace std;

int main()
{
	// This section declares the variables to be used for the program.
	float r ;
	float pi = 3.14159;
	float area;
	float circ;


	//This section prompts the user to enter the radius of the circle and stores it in float "r"
	cout << "This program calculates the area and circumference of a circle" << endl;
	cout << "Please enter the radius of the circle?" ;
	cin >> r ;

	//The equation to calculate area and circumference
	area = pi * r * r ;
	circ = 2 * pi * r ;

	//This section displays the output of the equations to the user.
	cout << "The area of your circle is " << area << endl ;
	cout << "The circumference of your circle is " << circ << endl ;

	return 0;
}

```

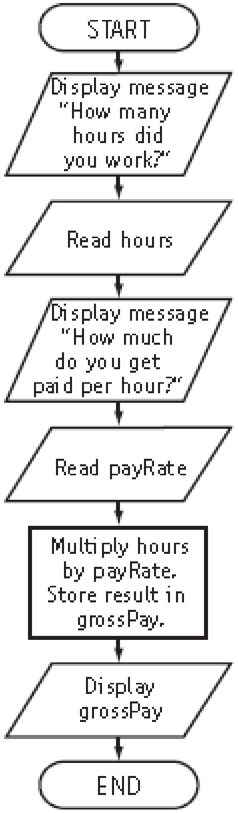
```
<terminated> circle1.exe [C/C++ Application] C:\Users\Admin\workspace\circle1\Debug\circle1.exe (9/29/13, 5:09 PM)
This program calculates the area and circumference of a circle
Please enter the radius of the circle?5.5
The area of your circle is 95.0331
The circumference of your circle is 34.5575
```

Gaddis Flowcharting Appendix (1)

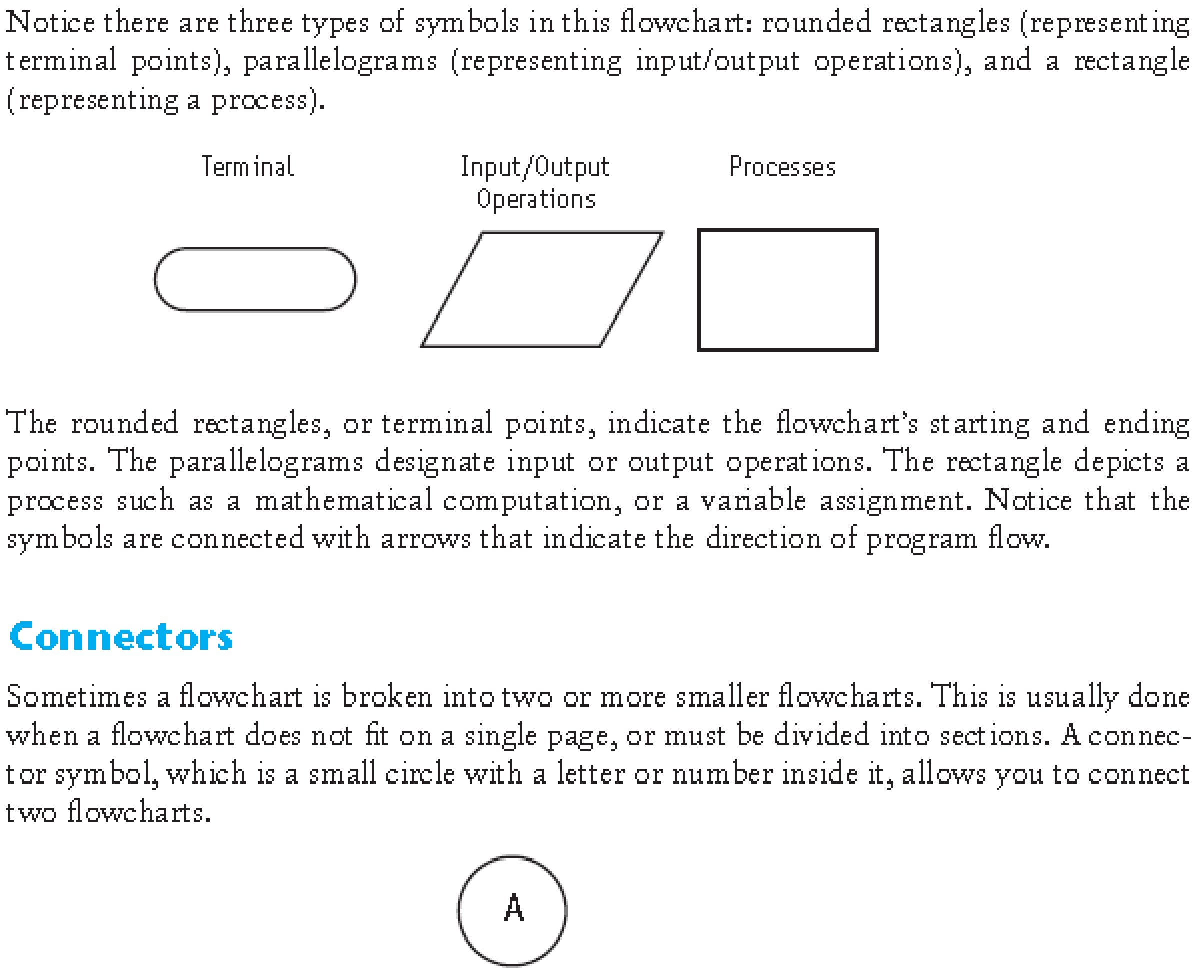


This appendix provides a brief introduction to flowcharting. It includes example flowcharts for programs that appear in Chapters 1 through 6.

A flowchart is a diagram that depicts the "flow" of a program. It contains symbols that represent each step in the program. The figure shown here is a flowchart for Program 1-1, the pay-calculating program in Chapter 1.


START
Display message "How many hours did you work?"
Read hours
Display message "How much do you get paid per hour?"
Read payRate
Multiply hours by payRate. Store result in grossPay.
Display grossPay
END

Gaddis Flowcharting Appendix (2)



Notice there are three types of symbols in this flowchart: rounded rectangles (representing terminal points), parallelograms (representing input/output operations), and a rectangle (representing a process).

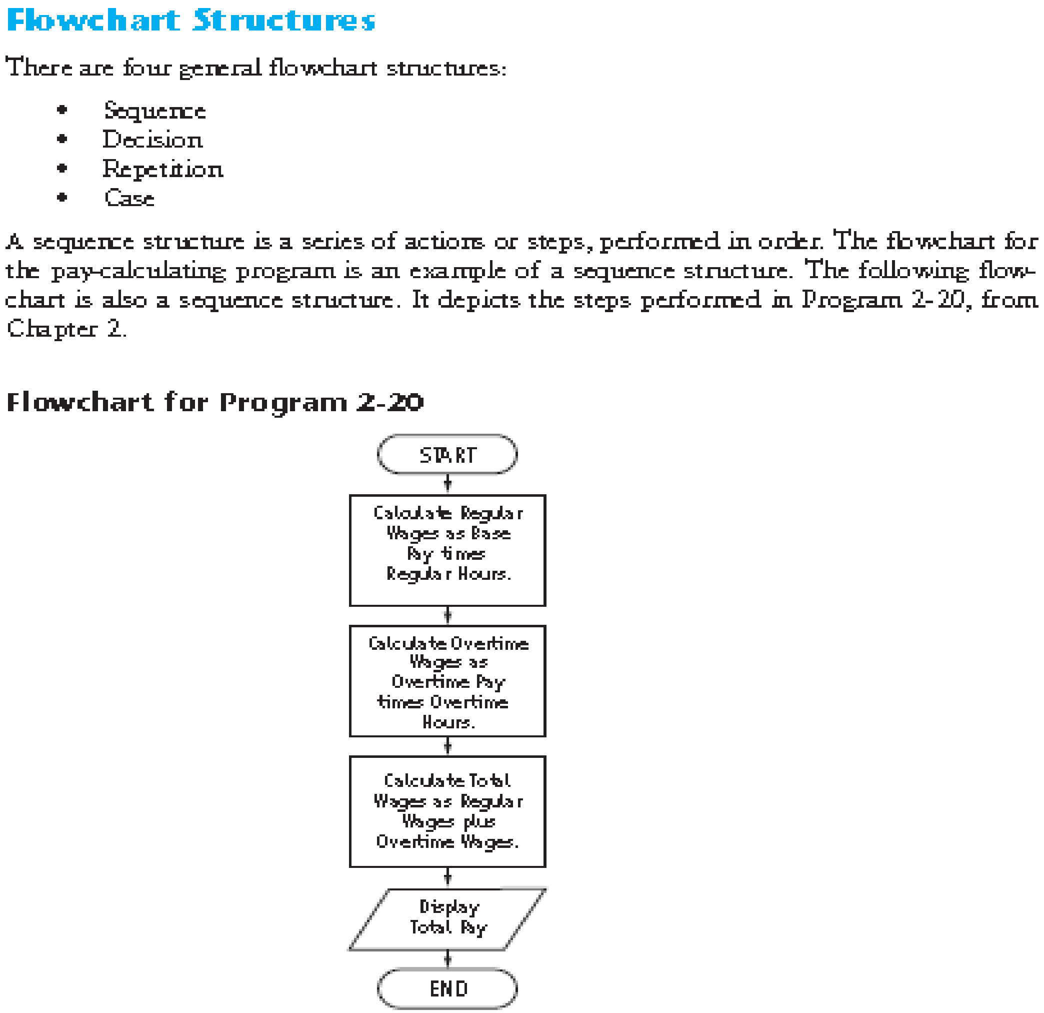
Terminal
Input/Output Operations
Processes

The rounded rectangles, or terminal points, indicate the flowchart's starting and ending points. The parallelograms designate input or output operations. The rectangle depicts a process such as a mathematical computation, or a variable assignment. Notice that the symbols are connected with arrows that indicate the direction of program flow.

Connectors

Sometimes a flowchart is broken into two or more smaller flowcharts. This is usually done when a flowchart does not fit on a single page, or must be divided into sections. A connector symbol, which is a small circle with a letter or number inside it, allows you to connect two flowcharts.

Gaddis Flowcharting Appendix (3)



Flowchart Structures

There are four general flowchart structures:

- Sequence
- Decision
- Repetition
- Case

A sequence structure is a series of actions or steps, performed in order. The flowchart for the pay-calculating program is an example of a sequence structure. The following flowchart is also a sequence structure. It depicts the steps performed in Program 2-20, from Chapter 2.

Flowchart for Program 2-20

START
Calculate Regular Wages as Base Pay times Regular Hours.
Calculate Overtime Wages as Overtime Pay times Overtime Hours.
Calculate Total Wages as Regular Wages plus Overtime Wages.
Display Total Pay
END

Arithmetic Operators

- Needed for performing calculations
- 3 kinds - unary, binary, and ternary operators:
  - unary (1 operand) `-5`
  - binary (2 operands) `13 - 7`
  - ternary (3 operands) `exp1 ? exp2 : exp3`

Binary Arithmetic Operators

SYMBOL	OPERATION	EXAMPLE	VALUE OF `ans`
`+`	addition	`ans = 7 + 3;`	10
`-`	subtraction	`ans = 7 - 3;`	4
`*`	multiplication	`ans = 7 * 3;`	21
`/`	division	`ans = 7 / 3;`	2
`%`	modulus	`ans = 7 % 3;`	1

/ - Division Operator

- Integer division if both operands are integers

```
cout << 13 / 5;     // displays 2
cout << 91 / 7;     // displays 13
```

- If either operand is floating point, the result is floating point

```
cout << 13 / 5.0;  // displays 2.6
cout << 91.0 / 7;  // displays 13.0
```

- If either operand is floating point but the result is stored in a variable, its data type results

```
int iRes = 13 / 5.0;      // stores 2
double dRes = 91.0 / 7;  // stores 13.0
```

% - Modulus Operator

- Used to find the remainder resulting from integer division

```
cout << 13 % 5;   // displays 3
```

- Both operands have to be integers

```
cout << 13 % 5.0; // error
```

Operator Precedence Order

- For computing operations, usual mathematical operator precedence is followed:
  - P E D M A S or P E M D A S
  - Parentheses before Exponents before (Division or Multiplication) before (Addition or Subtraction)
  - 2*4 + 2*5 = 8 + 10 = 18 → Multiplication before addition
  - 2 * (4 + 5) = 2 * 9 = 18 → Parenthesis resolved first
  - 2 * 5 * 5 = 50 → Straight multiplication
  - 2 * 5 ^ 2 = 2 * 25 = 50 → Exponent (^) first

    (note - ^ is not an exponent operator in C++ - just to demonstrate)

Order of Operations - 2

- With negation and modulus, evaluate in this order:
  - − (unary negation) → in order, left to right
  - `*   /   %` → in order, left to right
  - `+   −` → in order, left to right
- Associativity of operators is evaluated as:
  - ○ − (unary negation) → associates right to left
  - ○ `*,  /,  %,  +,  −` → associate left to right
  - ○ parentheses ( ) can be used to override the order of operations:

```
 2 + 2  *  2 − 2  = 4
(2 + 2) *  2 − 2  = 6
 2 + 2  * (2 − 2) = 2
(2 + 2) * (2 − 2) = 0
```

Algebraic Expressions

- Convert algebraic expressions based on operator precedence

- Multiplication requires an operator
  - `r = 2(3+5)` is written as `r = 2*(3+5);`

- C++ provides a function to perform exponents
  - $A = s^2$ is written as `A = pow(s,2);`
  - This function accepts and returns `float` or `double`
  - Include math library : `#include <cmath>`

- Parentheses help maintain order of operations
  - $m = \frac{y2 - y1}{x2 - x1}$ is written as `m = (y2 - y1) / (x2 - x1);`

Formatting Output

- Requires the `iomanip` library
- Control output display for numeric and string data
  - Size (width), Position, # of digits, Alignment

- `setw(x)` : print a field of at least x spaces
- `fixed` : use decimal notation
- `setprecision(x)` : print x significant digits after decimal
- `fixed & setprecision(x)` : print x digits after decimal
- `showpoint`: always print decimal point with trailing zeroes
- `left` : print values to be left justified (aligned)
- `right` : print values to be right justified (aligned)

Program 3-13

```
// This program displays three rows of numbers.
#include <iostream>
#include <iomanip>      // Required for setw
using namespace std;

int main()
{
   int num1 = 2897, num2 = 5,    num3 = 837,
       num4 = 34,   num5 = 7,    num6 = 1623,
       num7 = 390,  num8 = 3456, num9 = 12;

   // Display the first row of numbers
   cout << setw(6) << num1 << setw(6)
        << num2 << setw(6) << num3 << endl;

   // Display the second row of numbers
   cout << setw(6) << num4 << setw(6)
        << num5 << setw(6) << num6 << endl;

   // Display the third row of numbers
   cout << setw(6) << num7 << setw(6)
        << num8 << setw(6) << num9 << endl;
   return 0;
}
```

Program Output

```
  2897     5   837
    34     7  1623
   390  3456    12
```

Program 3-17

```
// This program asks for sales figures for 3 days. The total
// sales are calculated and displayed in a table.
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   double day1, day2, day3, total;

   // Get the sales for each day.
   cout << "Enter the sales for day 1: ";
   cin >> day1;
   cout << "Enter the sales for day 2: ";
   cin >> day2;
   cout << "Enter the sales for day 3: ";
   cin >> day3;

   // Calculate the total sales.
   total = day1 + day2 + day3;
```

```

    // Display the sales figures.
    cout << "\nSales Figures\n";
    cout << "-------------\n";
    cout << setprecision(2) << fixed;
    cout << "Day 1: " << setw(8) << day1 << endl;
    cout << "Day 2: " << setw(8) << day2 << endl;
    cout << "Day 3: " << setw(8) << day3 << endl;
    cout << "Total: " << setw(8) << total << endl;
    return 0;
}
```

Program Output with Example Input Shown in Bold

```
Enter the sales for day 1:  1321.87 [Enter]
Enter the sales for day 2:  1869.26 [Enter]
Enter the sales for day 3:  1403.77 [Enter]

Sales Figures
-------------
Day 1:   1321.87
Day 2:   1869.26
Day 3:   1403.77
Total:   4594.90
```