Fpt Aptech

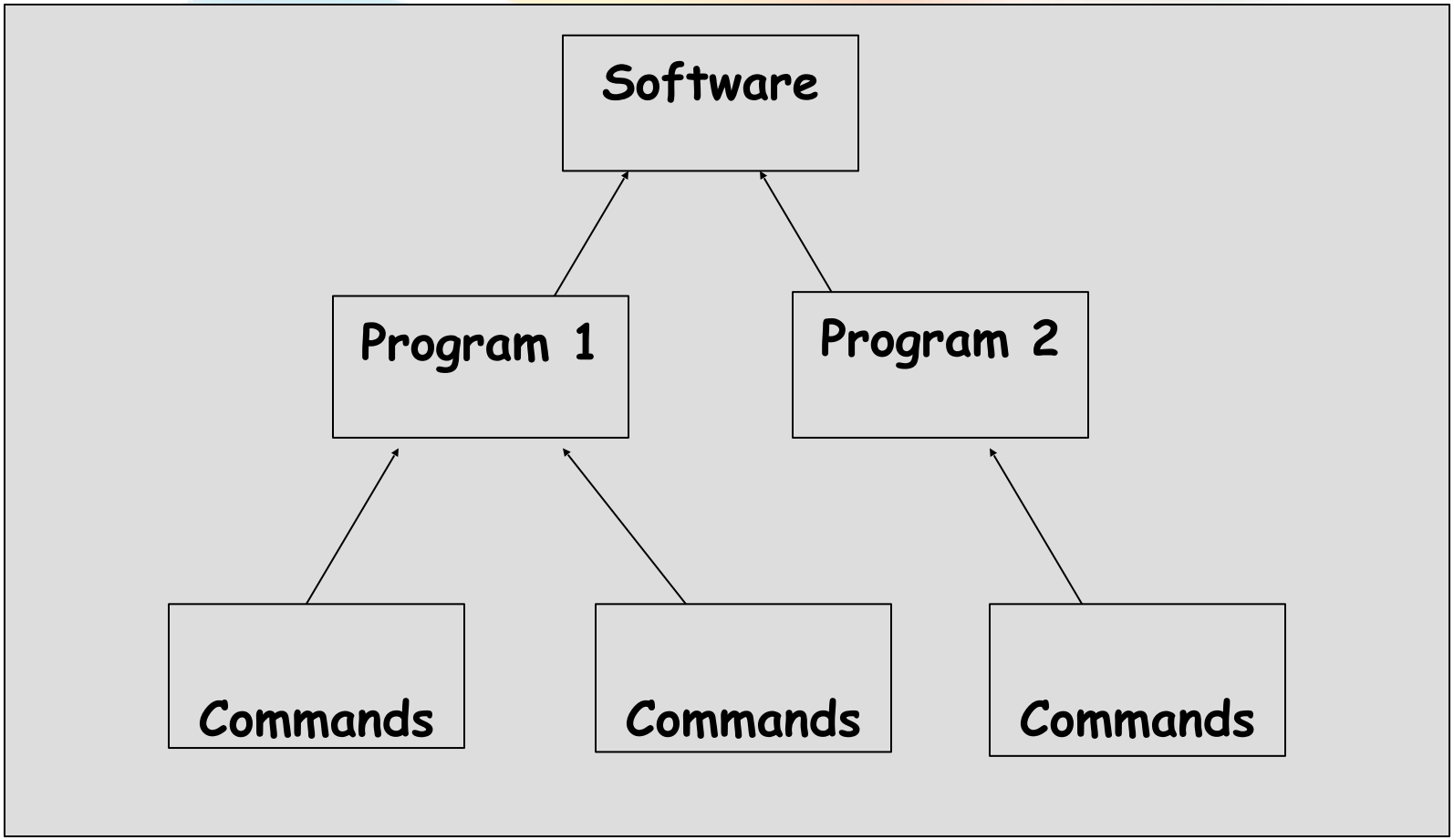
EPC, Session 1

Introduction to Programming

Objectives

- Differentiate between Command, Program and Software
- Explain the beginning of C
- Explain when and why is C used
- Discuss the C program structure
- Discuss algorithms
- Draw flowcharts
- List the symbols used in flowcharts

Software, Program and Command



The beginning of C

BPCL – Martin Richards

B – Ken Thompson

C – Dennis Ritchie

Bell Laboratories, Inc.

Application areas of C

- C was initially used for systems programming
- A system program forms a portion of the operating system of the computer or its support utilities
- Operating Systems, Interpreters, Editors, Assembly programs are usually called system programs
- The UNIX operating system was developed using C
- There are C compilers available for almost all types of PC's

Middle Level Language

High Level Language

C

Assembly Language

Structured Language

- C allows compartmentalization of code and data
- It refers to the ability to section off and hide all information and instructions, necessary to perform a specific task, from the rest of the program

```
do
{
    i = i + 1;
    .
    .
    .
} while (i < 40);
```

- Code can be compartmentalized in C by using functions or code blocks.

About C

- C has **32 keywords**
- These keywords combined **with a formal syntax** form a C programming language
- **Rules** to be followed for all programs written in C:
- All keywords are **lowercased**

```
main()
{
/* This is a sample Program*/
      int i,j;
      i=100;
      j=200;
        :
   }
```

- C is **case sensitive**, “do while” is different from “DO WHILE”
- **Keywords cannot** be used as a variable or function name

Functions

- C programs are divided into **units called functions**
- Irrespective of the number of functions in a program, the operating system always passes control to the **main()** when a C program is executed.
- The **function name** is always followed be **parentheses**.
- The parentheses **may or not contain parameters**.

main()

Delimiters

- The function definition is followed by an open curly brace ({)
- The curly brace signals the beginning of the function
- A closing curly brace (}) after the codes, in the function, indicate the end of the function

Delimiters { ... }

Terminator

- A statement in C is terminated with a semicolon
- A carriage return, whitespace, or a tab is not understood by the C compiler
- A statement that does not end in a semicolon is treated as an erroneous line of code in C

Statement

Comment

- Comments are usually written to describe the task of a particular command, function or an entire program
- The compiler ignores comments.
- There are two way to insert comments:
  - Single line:

```
//Comments go here
```

  - Multiline:

```
/*
Comments go here        and here
*/
```

A Simple C Program

Printing a line of text

```c
/* Fig. 2.1: fig02_01.c
   A first program in C */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
   printf( "Welcome to C!\n" );

   return 0; /* indicate that program ended successfully */
} /* end function main */
```

```
Welcome to C!
```

Escape Sequence

Escape sequence	Description
\n	Newline. Position the cursor at the beginning of the next line.
\t	Horizontal tab. Move the cursor to the next tab stop.
\a	Alert. Sound the system bell.
\\	Backslash. Insert a backslash character in a string.
\"	Double quote. Insert a double-quote character in a string.

A Simple C Program

```c
/* Fig. 2.3: fig02_03.c
   Printing on one line with two printf statements */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
   printf( "Welcome " );
   printf( "to C!\n" );

   return 0; /* indicate that program ended successfully */
} /* end function main */
```

Fig. 2.3 | Printing on one line with two `printf` statements. (Part 1 of 2.)

A Simple C Program

```
Welcome to C!
```

Fig. 2.3 | Printing on one line with two `printf` statements. (Part 2 of 2.)

A Simple C Program

```c
/* Fig. 2.4: fig02_04.c
   Printing multiple lines with a single printf */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
   printf( "Welcome\nto\nC!\n" );

   return 0; /* indicate that program ended successfully */
} /* end function main */
```

```
Welcome
to
C!
```

Fig. 2.4 | Printing multiple lines with a single `printf`.

HelloC 〉 My Mac | Finished running HelloC : HelloC

HelloC 〉 HelloC 〉 main.c 〉 main

- HelloC
  - HelloC
    - main.c
  - Products

```c
//
//  main.c
//  HelloC
//
//  Created by Dang Kim Thi on 18/08/2018.
//  Copyright © 2018 Dang Kim Thi. All rights reserved.
//

#include <stdio.h>

int main(int argc, const char * argv[]) {
    // insert code here...
    printf("Hello\n, World!");
    printf("ahihi\n");
    return 0;
}
```

```
Hello
, World!ahihi
Program ended with exit code: 0
```

All Output | Filter

Identity and Type

Name: main.c

Type: Default - C Source

Location: Relative to Group

main.c

Full Path: /Users/thidk/Library/Mobile Documents/com~apple~CloudDocs/Documents/Aptech/1.SEM1/1.Logic Building and Elementary Programming/DemoLive/Demo_func/Calculator/HelloC/HelloC/main.c

On Demand Resource Tags

Only resources are taggable

Target Membership

HelloC

Cocoa Touch Class - A Cocoa Touch class

UI Test Case Class - A class implementing a unit test

Unit Test Case Class - A class implementing a unit test

Another Simple C Program

```c
/* Fig. 2.5: fig02_05.c
   Addition program */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
   int integer1; /* first number to be input by user  */
   int integer2; /* second number to be input by user */
   int sum; /* variable in which sum will be stored */

   printf( "Enter first integer\n" ); /* prompt */
   scanf( "%d", &integer1 ); /* read an integer */

   printf( "Enter second integer\n" ); /* prompt */
   scanf( "%d", &integer2 ); /* read an integer */
```

Fig. 2.5 | Addition program. (Part 1 of 2.)

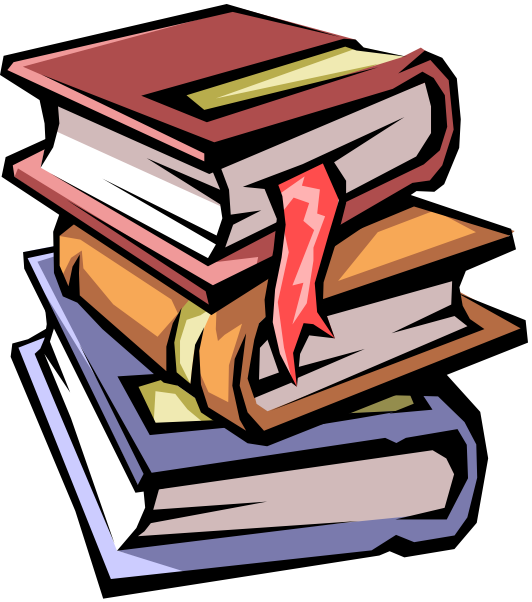
Another Simple C Program

```c

    sum = integer1 + integer2; /* assign total to sum */

    printf( "Sum is %d\n", sum ); /* print sum */

    return 0; /* indicate that program ended successfully */
 } /* end function main */
```

```
Enter first integer
45
Enter second integer
72
Sum is 117
```

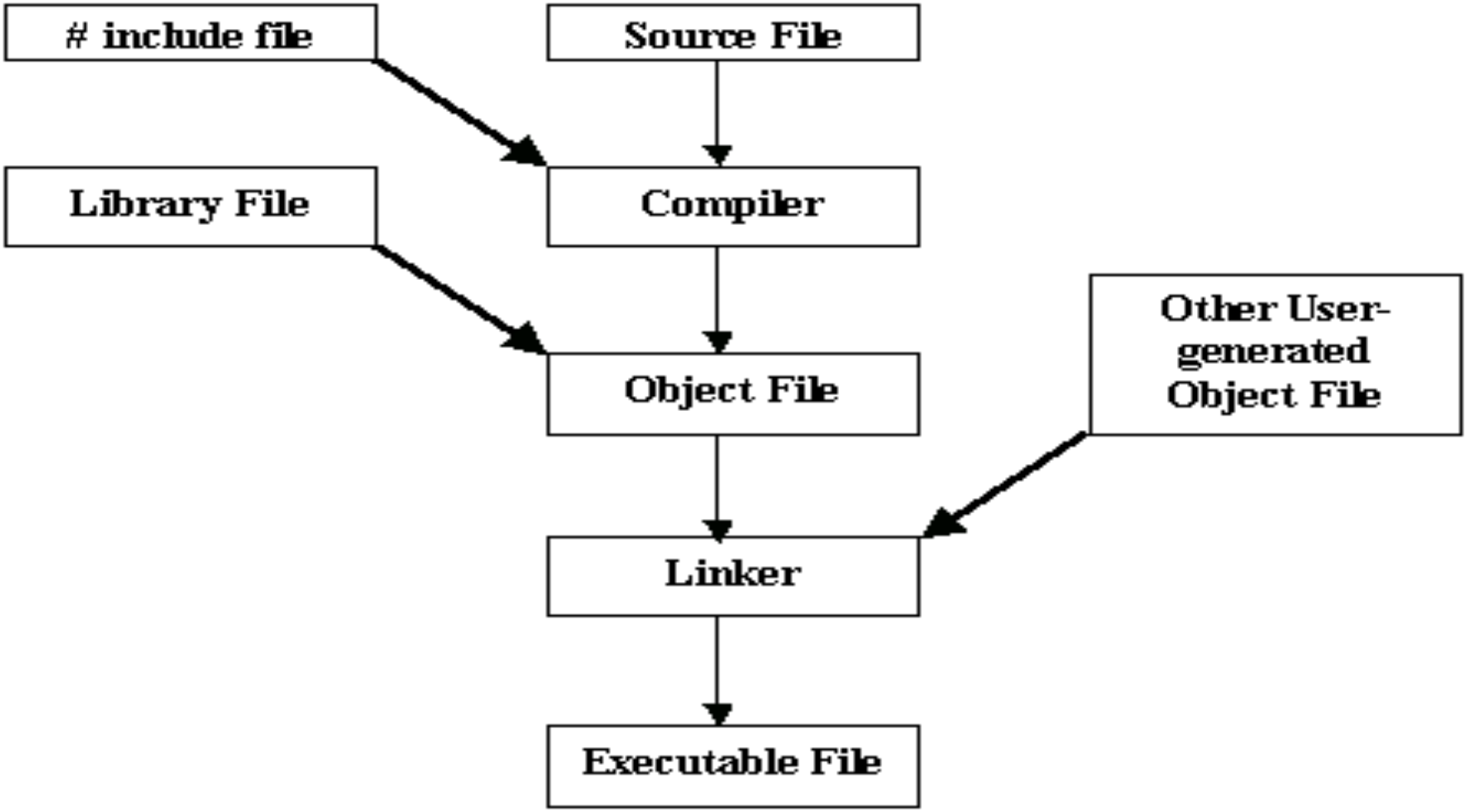
Fig. 2.5 | Addition program. (Part 2 of 2.)

The C Library



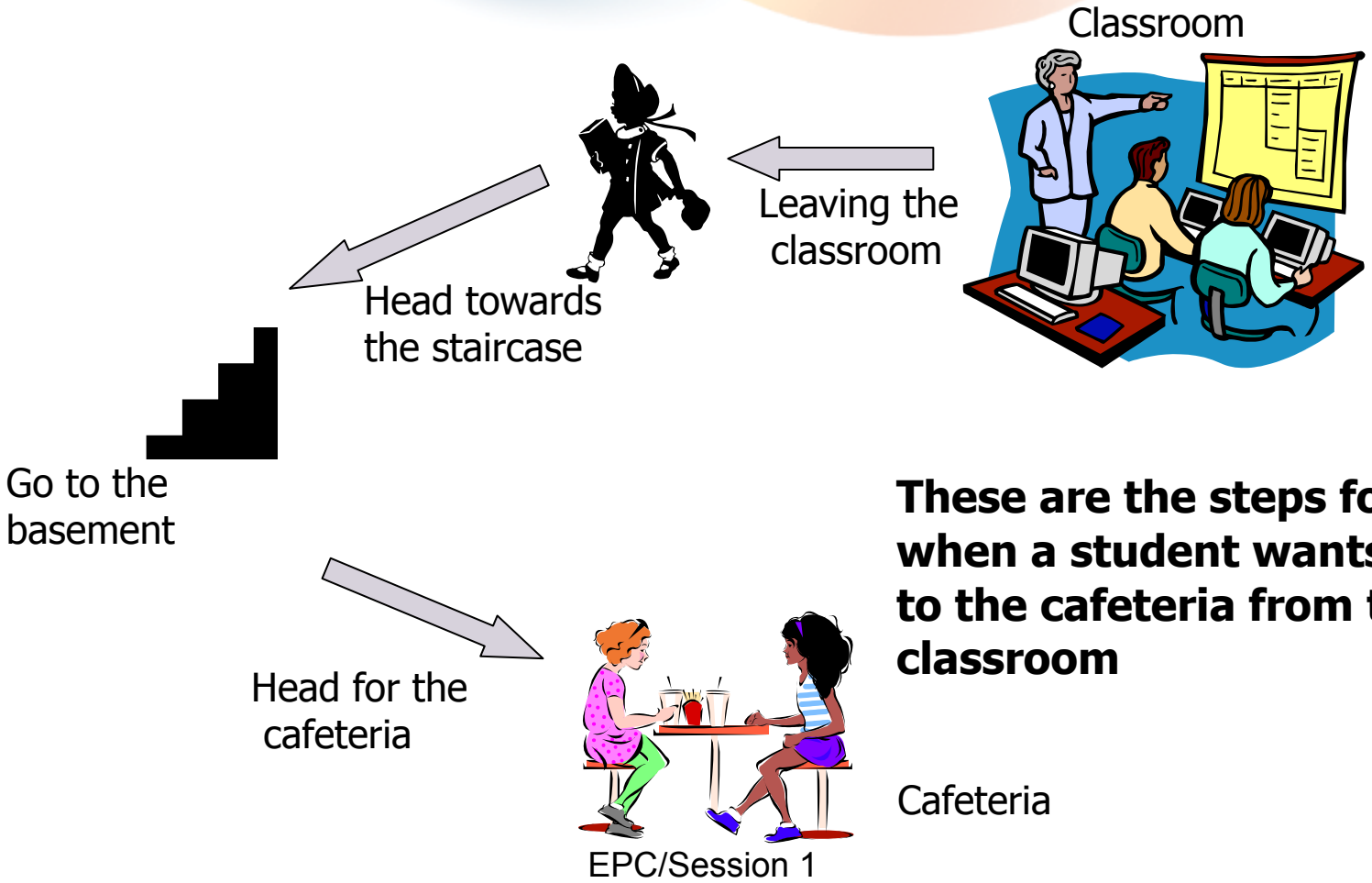
- All C compilers come with a standard library of functions
- A function written by a programmer can be placed in the library and used when required
- Some compilers allow functions to be added in the standard library
- Some compilers require a separate library to be created

Compiling & Running A Program



The Programming Approach to Solving Problems

Algorithm is a set of steps that are performed to solve a problem. The example below describes an algorithm:



These are the steps followed when a student wants to go to the cafeteria from the classroom

Solving a Problem

In order to solve a problem

- Understand the problem clearly
- Gather the relevant information
- Process the information
- Arrive at the solution

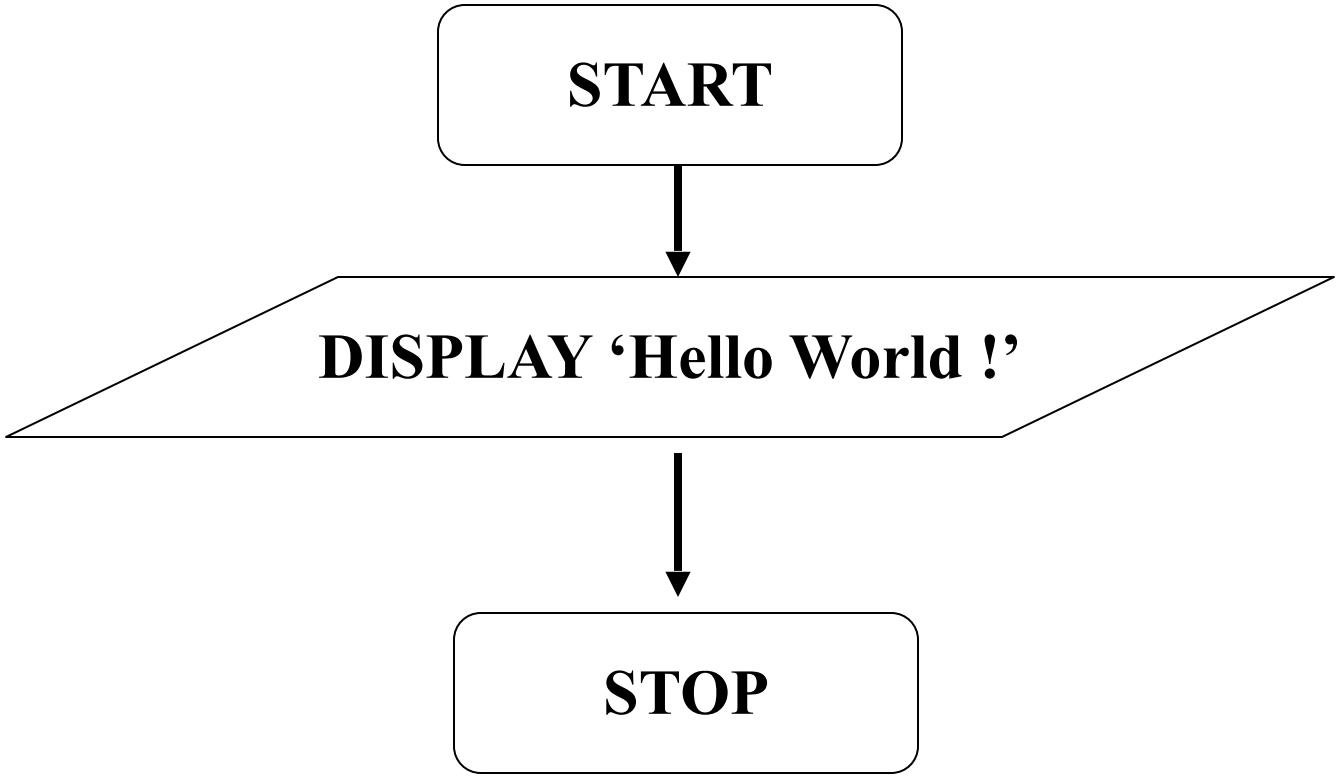
Pseudocode

- Is not actual code.
- Is a method of algorithm - writing which uses a standard set of words which makes it resemble code
- Each pseudocode starts with a BEGIN
- To show some value , the word DISPLAY is used
- The pseudocode finishes with an

```
BEGIN
DISPLAY 'Hello World !'
END
```

Flowcharts

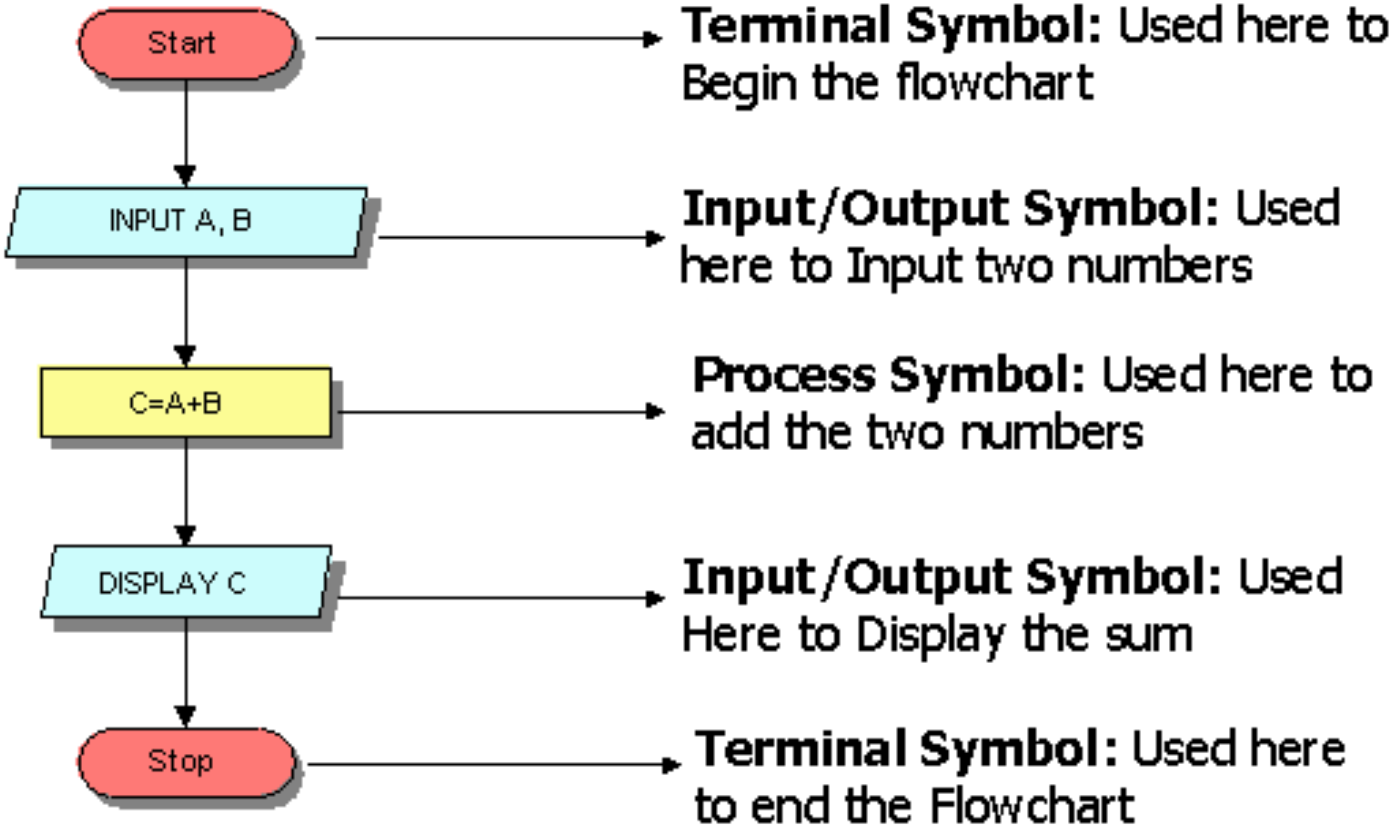
It is a graphical representation of an algorithm



The Flowchart Symbol

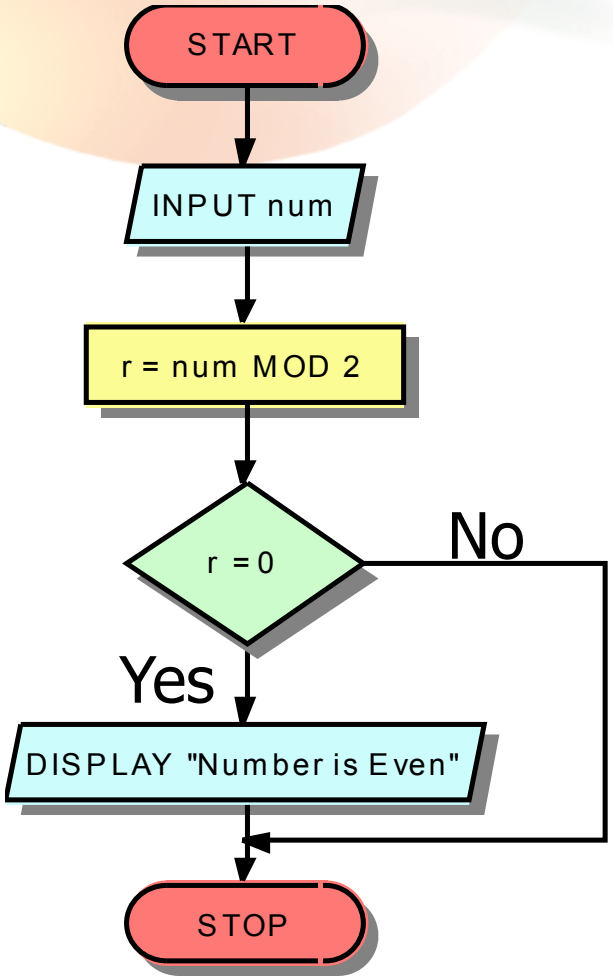
Symbol	Description
	Start or End of the Program
	Computational Steps
	Input / Output instructions
	Decision making & Branching
	Connectors
	Flow Line

Flowchart to add two numbers



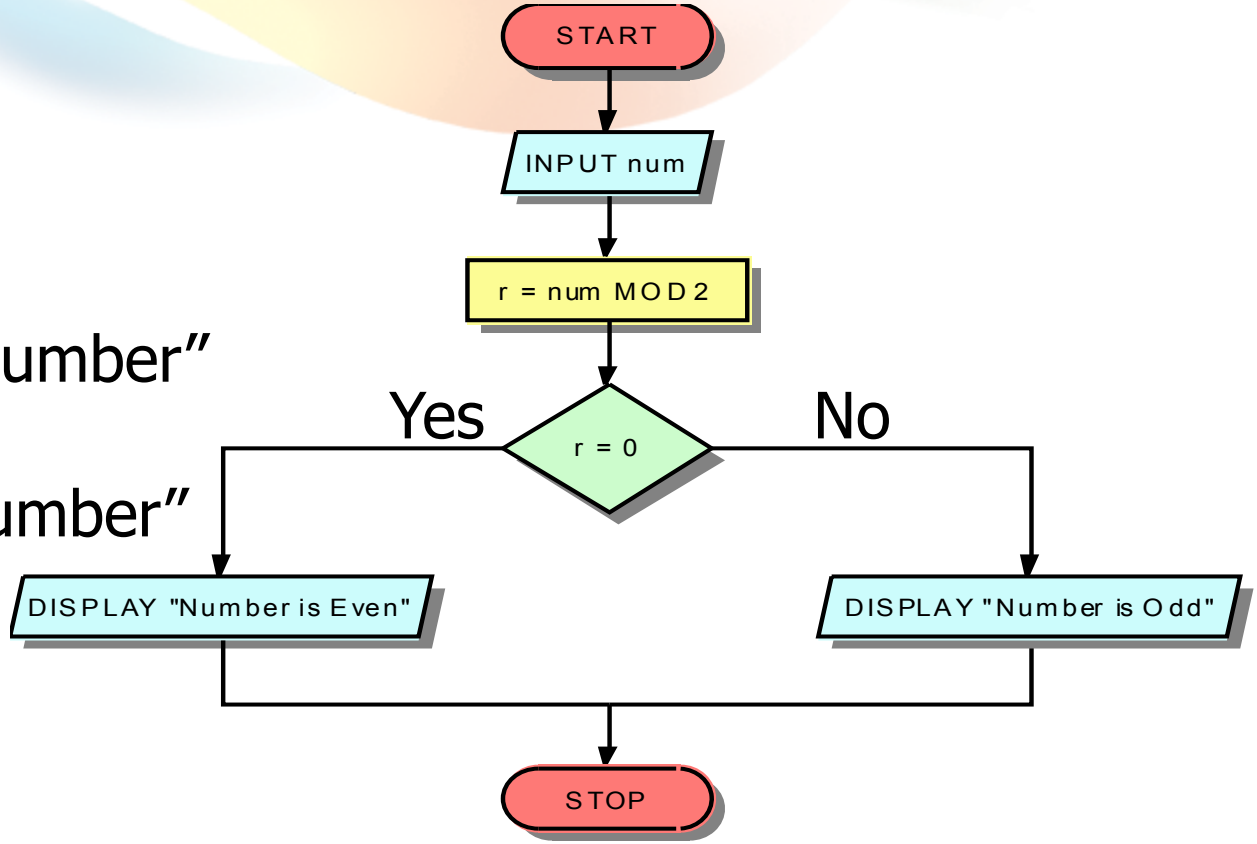
The IF Construct

BEGIN
INPUT num
r = num MOD 2
IF r=0
Display “Number is even”
END IF
END



The IF-ELSE Construct

```
BEGIN
INPUT num
r=num MOD 2
IF r=0
   DISPLAY "Even Number"
ELSE
   DISPLAY "Odd Number"
END IF
END
```



DEMO IF ELSE

```c
/* Fig. 2.13: fig02_13.c
   Using if statements, relational
   operators, and equality operators */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
   int num1; /* first number to be read from user  */
   int num2; /* second number to be read from user */

   printf( "Enter two integers, and I will tell you\n" );
   printf( "the relationships they satisfy: " );

   scanf( "%d%d", &num1, &num2 ); /* read two integers */

   if ( num1 == num2 ) {
      printf( "%d is equal to %d\n", num1, num2 );
   } /* end if */

   if ( num1 != num2 ) {
      printf( "%d is not equal to %d\n", num1, num2 );
   } /* end if */

   if ( num1 < num2 ) {
      printf( "%d is less than %d\n", num1, num2 );
   } /* end if */

```

Fig. 2.13 | Using `if` statements, relational operators, and equality operators. (Part 1 of 2.)

```
    if ( num1 > num2 ) {
       printf( "%d is greater than %d\n", num1, num2 );
    } /* end if */

    if ( num1 <= num2 ) {
       printf( "%d is less than or equal to %d\n", num1, num2 );
    } /* end if */

    if ( num1 >= num2 ) {
       printf( "%d is greater than or equal to %d\n", num1, num2 );
    } /* end if */

    return 0; /* indicate that program ended successfully */
} /* end function main */
```

```
Enter two integers, and I will tell you
the relationships they satisfy: 3 7
3 is not equal to 7
3 is less than 7
3 is less than or equal to 7
```

```
Enter two integers, and I will tell you
the relationships they satisfy: 12 12
22 is not equal to 12
22 is greater than 12
22 is greater than or equal to 12
```

```
Enter two integers, and I will tell you
the relationships they satisfy: 7 7
7 is equal to 7
7 is less than or equal to 7
7 is greater than or equal to 7
```

Fig. 2.13 | Using `if` statements, relational operators, and equality operators. (Part 2 of 2.)

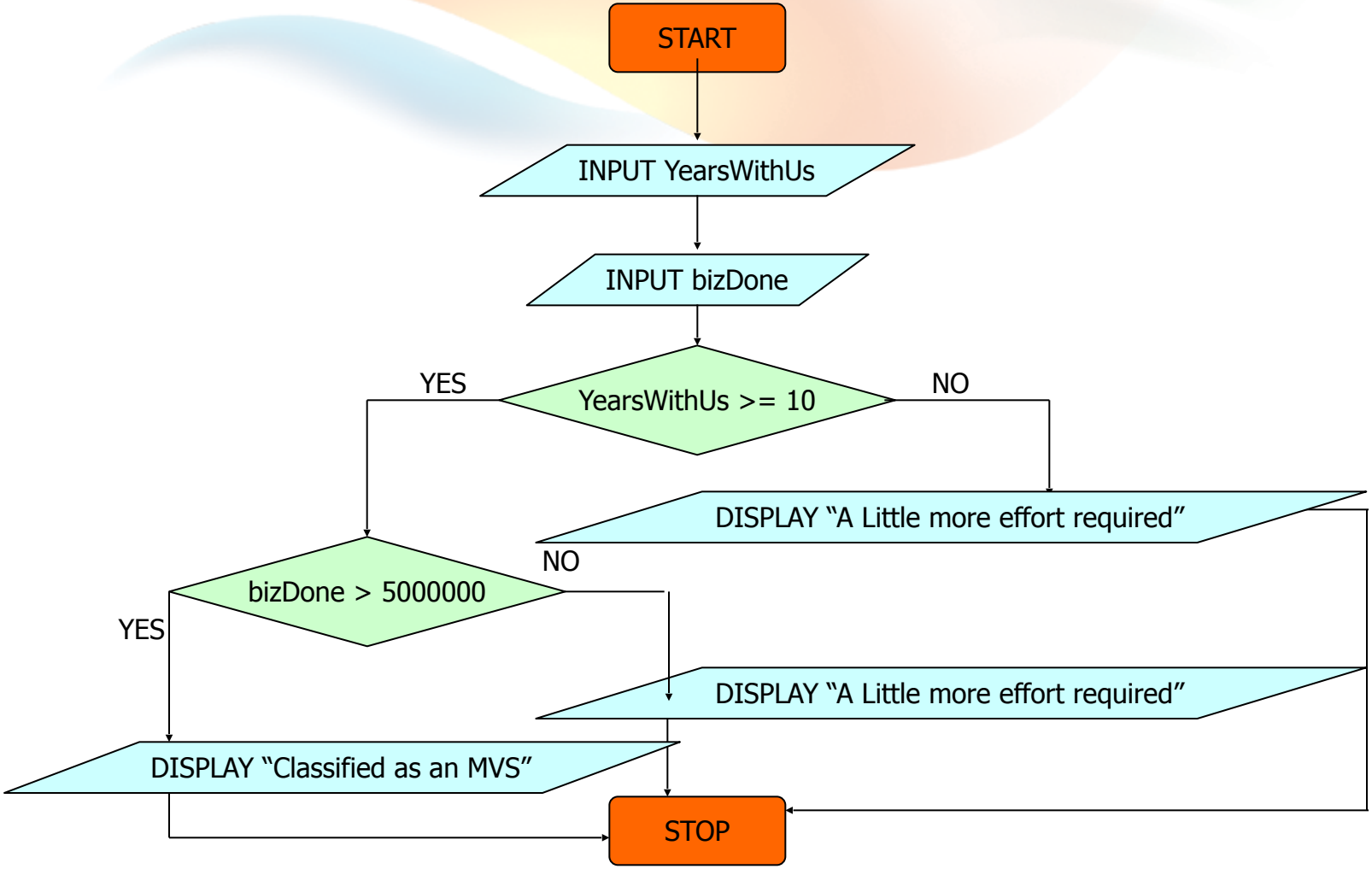
Multiple criteria using AND/OR

```
BEGIN
INPUT yearsWithUs
INPUT bizDone
IF yearsWithUs >= 10 AND bizDone >=5000000
        DISPLAY "Classified as an MVS"
ELSE
        DISPLAY "A little more effort required!"
END IF
END
```

Nested IFs-1

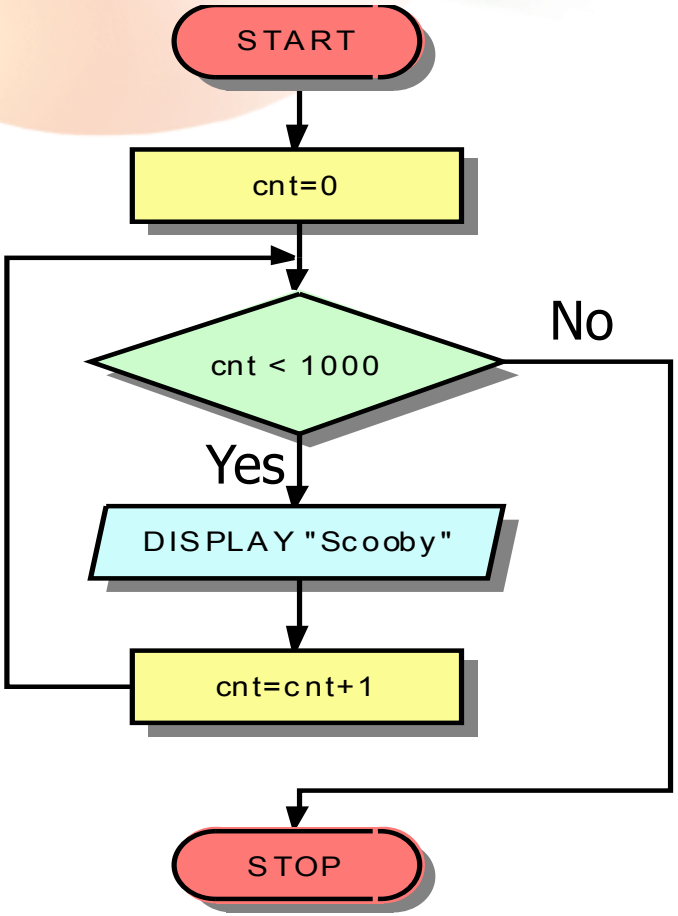
```
BEGIN
INPUT yearsWithUs
INPUT bizDone
IF yearsWithUs >= 10
IF bizDone >=5000000
        DISPLAY "Classified as an MVS"
           ELSE
                DISPLAY "A little more effort required!"
END IF
ELSE
        DISPLAY "A little more effort required!"
END IF
END
```

Nested IFs-2


START
INPUT YearsWithUs
INPUT bizDone
YEARS
YearsWithUs >= 10
NO
DISPLAY "A Little more effort required"
NO
bizDone > 5000000
YES
DISPLAY "A Little more effort required"
DISPLAY "Classified as an MVS"
STOP

Loops

```
BEGIN
cnt=0
WHILE (cnt < 1000)
DO
        DISPLAY "Scooby"
        cnt=cnt+1
END DO
END
```



Exercise

```
*********          ***          *          *
*       *        *     *       ***        * *
*       *       *       *     *****      *   *
*       *       *       *       *       *     *
*       *       *       *       *      *       *
*       *       *       *       *      *       *
*       *       *       *       *       *     *
*       *        *     *        *        * *
*********          ***          *          *
```

DEMO LOOP

```c
/* Fig. 3.6: fig03_06.c
   Class average program with counter-controlled repetition */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
   int counter; /* number of grade to be entered next */
   int grade; /* grade value */
   int total; /* sum of grades input by user */
   int average; /* average of grades */

   /* initialization phase */
   total = 0; /* initialize total */
   counter = 1; /* initialize loop counter */

   /* processing phase */
   while ( counter <= 10 ) { /* loop 10 times */
      printf( "Enter grade: " ); /* prompt for input */
      scanf( "%d", &grade ); /* read grade from user */
      total = total + grade; /* add grade to total */
      counter = counter + 1; /* increment counter */
   } /* end while */

   /* termination phase */
   average = total / 10; /* integer division */

   printf( "Class average is %d\n", average ); /* display result */
   return 0; /* indicate program ended successfully */
} /* end function main */
```

Fig. 3.6 | C program and sample execution for the class average problem with counter-controlled repetition. (Part 1 of 2.)

OUTPUT

```
Enter grade: 98
Enter grade: 76
Enter grade: 71
Enter grade: 87
Enter grade: 83
Enter grade: 90
Enter grade: 57
Enter grade: 79
Enter grade: 82
Enter grade: 94
Class average is 81
```

Fig. 3.6 | C program and sample execution for the class average problem with counter-controlled repetition. (Part 2 of 2.)

Exercise

```
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
```

```
PPPPPPPPP
    P   P
    P   P
    P   P
     P P

  JJ
 J
J
 J
  JJJJJJJ

DDDDDDDDD
D       D
D       D
 D     D
  DDDDD
```

Summary

- software is a set of programs.
- A Program is a set of instructions.
- Code blocks, form a base of any C program.
- An Algorithm is a logical and concise list of steps to solve a problem
- A pseudo code is a representation of an algorithm in language that resembles code
- A flowchart is a diagrammatic representation of an algorithm
- The basic selection construct is an ‘IF’ construct
- The iterative or looping constructs is necessary to repeat certain steps