CSE 06131223 ♦ CSE 06131224

Structured Programming

Lecture 4

Computer Programming Fundamentals (3)

Prepared by

Md. Mijanur Rahman, Prof. Dr.
Dept. of Computer Science and Engineering
Jatiya Kabi Kazi Nazrul Islam University, Bangladesh
www.mijanrahman.com

Contents

Computer Programming Fundamentals

- Introduction
- Software and Programming Language
- Types of Programming Languages
- Commonly Used Programming Languages
- Program Development Life Cycle
- Types of Errors in Programs
- Algorithm and Flowchart
- **Control Structures**
- **Pseudo Code**
- **Programming Paradigms**

Control Structures

- **The logic of a program may not always be a linear sequence of statements to be executed in that order.** The logic of the program may require execution of a statement based on a decision. It may repetitively execute a set of statements unless some condition is met.
- ***Control structures* specify the statements to be executed and the order of execution of statements.** There are three kinds of control structures:
  1. **Sequence**
  2. **Selection**
  3. **Iteration**

Control Structures

- **Sequence:**
- The most common convention in programs and in everyday life is that we follow the instructions, starting at the top and working down the list, doing the instructions in the order given (this is a sequence of instructions)
- The sequence of instructions ***Lights – Camera – Action*** not only tells a film crew what to do but also the order in which to do things.

Control Structures

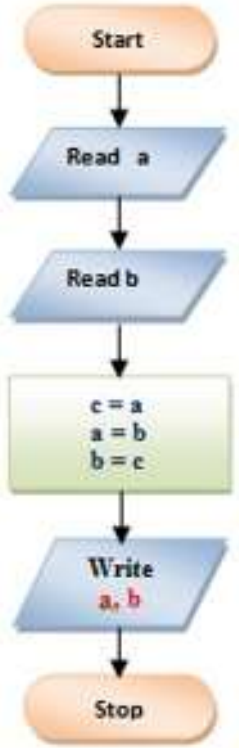
- **Sequence:**

1. Start
2. Read two values into two variables a, b
3. Declare third variable, c

   c = a

   a = b

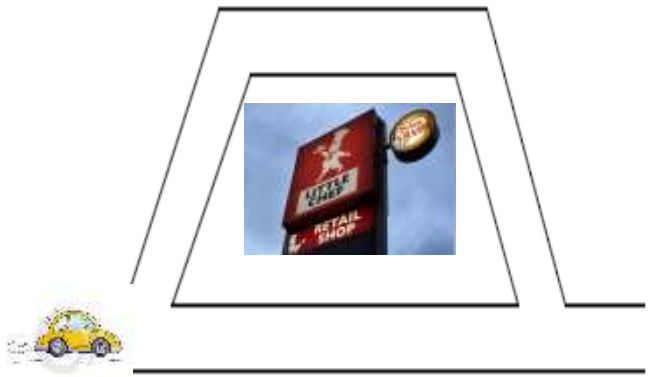
   b = c
4. Print or display a, b
5. Stop




```c
#include<stdio.h>

int main()
{

    int a, b, c;
    printf("Enter value of a:");
    scanf("%d", &a);

    printf("Enter value of b:");
    scanf("%d", &b);

    c = a;
    a = b;
    b = c;

    printf("Values of a & b after swapping: ");
    printf("a = %d\n", a);
    printf("b = %d", b);

    return 0;
}
```

Control Structures

- **Selection (If…Else):**
- Sequencing is not the only way instructions can be ordered. You can alter the sequence of instructions depending on the situation. This is known as **selection** or **branching**.

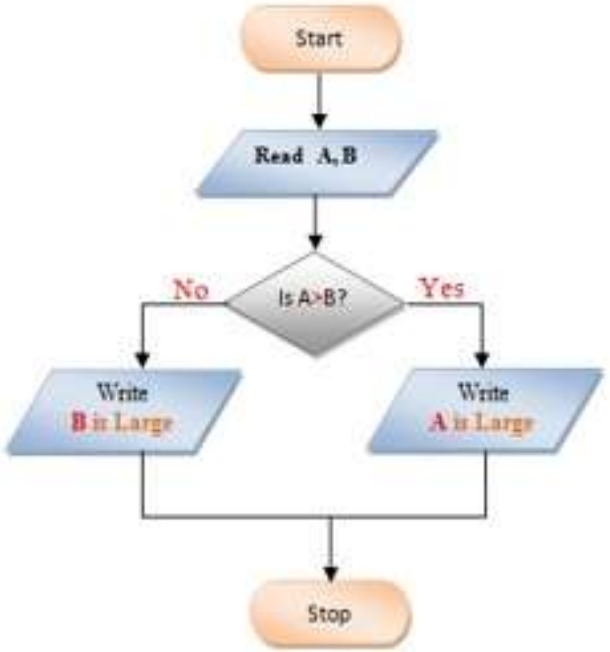


- Say, a motorway has a service station. A driver, drives down the motorway. On seeing the sign for the Service Station, she asks herself the question:
- "Do I need a break".
- If the answer is yes, she pulls in and has a coffee. On finishing the coffee she pulls back onto the motorway, using the exit slip road.
- If the answer is no, she just drives on and performs no action.
- In either case she arrives at the same point to continue her journey.

Control Structures

- **Selection (If…Else):**

1. Start
2. Read A,B
3. If A > B then
   Print A is large
   else
   Print B is large
4. Stop




```c
#include<stdio.h>

int main()
{

    int A, B;

    printf("Enter values of A, B: ");
    scanf("%d %d", &A, &B);

    if (A>B)
    printf("A is Larger");
    else
    printf("B is Larger");

    return 0;
}
```

Control Structures

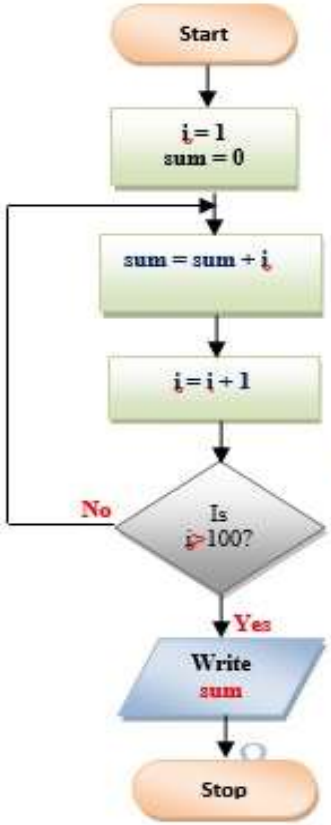
- **Iteration (Loop):**
- With fixed iteration we know in advance the number of times an instruction needs to be repeated.

- At school as a punishment you might have to write lines: writing the same thing over and over again.
- “Write out 100 times, ‘I must not throw chewing gum at the teacher’ ”
- “Write out 30 times, ‘I must not break up my desk and pass the bits out of the window behind the teacher’s back’ ”,

Control Structures

- **Iteration (Loop):**

1. Start
2. Initialize count i = 1, sum = 0
3. sum = sum + i
4. Increment i by 1
5. Repeat steps 3 & 4 until i > 100
6. Print sum
7. Stop




```c
#include<stdio.h>

int main()
{
    int i, sum;

    sum =0;
    for(i=1; i<101;i++)
    {
        sum = sum + i;
    }

    printf("Sum of integers from 1 to 100 is: %d", sum);

    return 0;
}
```

Pseudocode

- **Pseudocode is an artificial and informal language that helps programmers develop algorithms. Pseudocode is a "text-based" detail (algorithmic) design tool.** It consists of short, readable and formally-styled English language used for explaining an algorithm.
- Pseudocode does not include details like variable declarations, subroutines etc. Pseudo code is a short-hand way of describing a computer program.
- **Preparing a Pseudocode:** Pseudocode is written using structured English. In a pseudo code, some terms are commonly used to represent the various actions.
- For example, for inputting data the terms may be (INPUT, GET, READ), for outputting data (OUTPUT, PRINT, DISPLAY), for calculations (COMPUTE, CALCULATE), for incrementing (INCREMENT), in addition to words like ADD, SUBTRACT, INITIALIZE used for addition, subtraction, and initialization, respectively.

Pseudocode: Control Structures

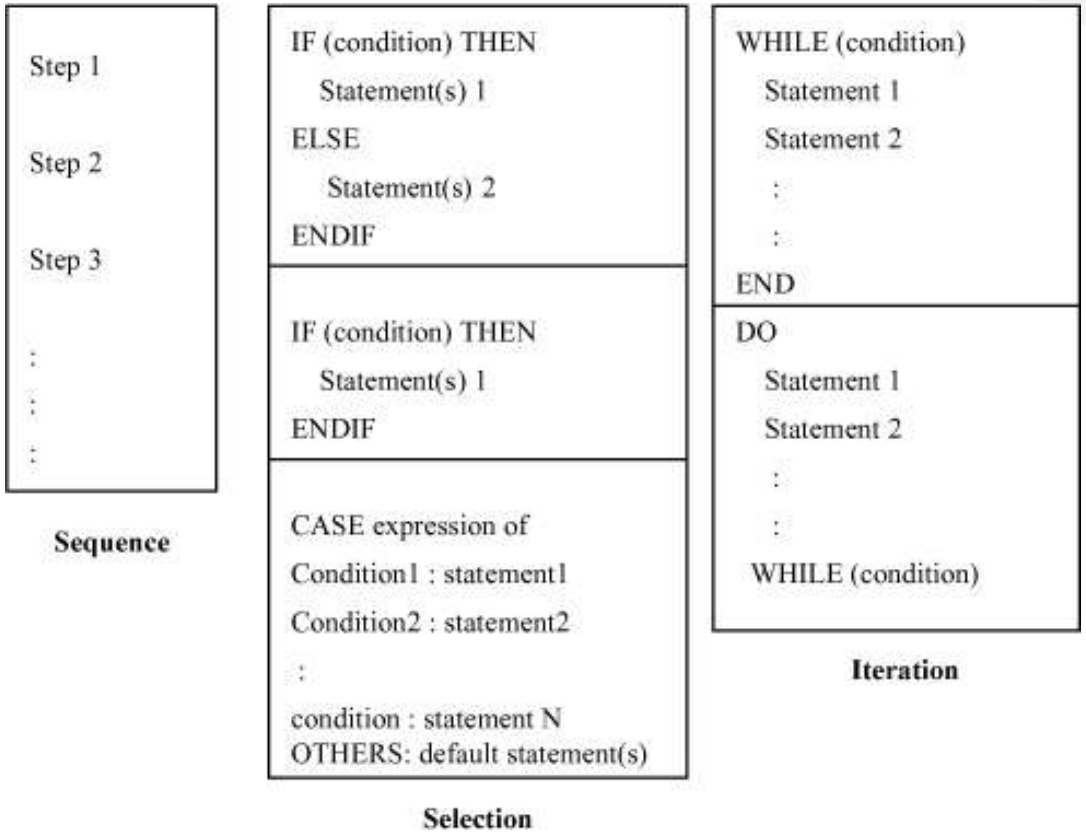


Fig: The different pseudocode structures

- The control structures—**sequence**, **selection**, and **iteration** are also used while writing the pseudocode.

- The ***sequence structure*** is simply a sequence of steps to be executed in linear order.
- There are two main ***selection constructs***—if-statement and case statement.
- WHILE and DO-WHILE are the two iterative statements. The WHILE loop and the DO-WHILE loop, both execute while the condition is true.

Algorithm vs Pseudocode

- **Algorithm:** It's an organized logical sequence of the actions or the approach towards a particular problem. A programmer implements an algorithm to solve a problem. Algorithms are expressed using natural verbal but somewhat technical annotations.

- **Pseudocode:** It's simply an implementation of an algorithm in the form of annotations and informative text written in plain English. It has no syntax like any of the programming language and thus can't be compiled or interpreted by the computer.

Advantages of Pseudocode

- **Followings are the major advantages of Pseudocode:**
- Improves the readability of any approach. It’s one of the best approaches to start implementation of an algorithm.
- Acts as a bridge between the program and the algorithm or flowchart. Also works as a rough documentation, so the program of one developer can be understood easily when a pseudo code is written out.
- The main goal of a pseudo code is to explain what exactly each line of a program should do, hence making the code construction phase easier for the programmer.

Basic Rules in Writing Pseudocode

- **Following are the basic rules before writing pseudocode:**
  - Write only one statement per line.
  - Write what you mean, not how to program it.
  - Give proper indentation to show hierarchy and make code understandable.
  - Make the program as simple as possible.
  - Conditions and loops must be specified well, i.e., begun and ended explicitly.

Pseudocode Examples

- Write a pseudocode to find the largest of two numbers.

```
1
2  BEGIN
3
4    NUMERIC nNum1,nNum2
5    DISPLAY "ENTER THE FIRST NUMBER : "
6    INPUT nNum1
7
8    DISPLAY "ENTER THE SECOND NUMBER : "
9    INPUT nNum2
10
11   IF nNum1 > nNum2
12     DISPLAY nNum1 + " is larger than "+ nNum2
13   ELSE
14    DISPLAY nNum2 + " is larger than " + nNum1
15
16 END
17
```

Pseudocode Examples

- Write a pseudocode to find the sum of three numbers.

```
1
2  begin
3      numeric nNum1,nNum2,nNum3,nSum
4      display "ENTER THE FIRST NUMBER : "
5      accept nNum1
6      display "ENTER THE SECOND NUMBER : "
7      accept nNum2
8      display "ENTER THE THIRD NUMBER : "
9      accept nNum3
10     nSum=nNum1+nNum2+nNum3
11     display "SUM OF ALL THREE NUMBERS : " nSum
12 end
13
```

Pseudocode Examples

- Write a pseudocode to find the perimeter of rectangle.

```
1
2  begin
3      numeric nLen,nBrd,nAre
4      display "ENTER THE LENGTH OF RECGTANGLE : "
5      accept nLen
6      display "ENTER THE BREADTH OF RECTANGLE : "
7      accept nBrd
8      nAre=2*(nLen+nBrd)
9      display "PERIMETER OF RECTANGLE : " nAre
10 end
11
```

Pseudocode Examples

- Write a pseudocode to check whether the entered number is even or odd.

```
1
2  begin
3      numeric nNum
4      display "ENTER A NUMBER : "
5      accept nNum
6      if(nNum%2==0)
7      begin
8          display "EVEN"
9      end
10     else
11     begin
12         display "ODD"
13     end
14 end
15
```

Pseudocode Examples

- Write a pseudocode to find the greatest of three numbers.

```
1
2  begin
3      numeric nNum1, nNum2, nNum3
4      display "ENTER THE FIRST NUMBER : "
5      accept nNum1
6      display "ENTER THE SECOND NUMBER : "
7      accept nNum2
8      display "ENTER THE THIRD NUMBER : "
9      accept nNum3
10     Num=nNum1
11     if(Num<nNum2)
12     begin
13         Num=nNum2
14     end
15     if(Num<nNum3)
16     begin
17         Num=nNum3
18     end
19     display "GREATEST ONE : " Num
20 end
21
```

Pseudocode Examples

- Write a pseudocode to check whether the entered year is a leap year or not.

```
1
2  begin
3      numeric nYear
4      display "ENTER THE YEAR "
5      accept nYear
6      if(nYear%4==0)
7      begin
8          display "THIS IS A LEAP YEAR"
9      end
10     else
11     begin
12         display "THIS IS NOT A LEAP YEAR"
13     end
14 end
15
```

Pseudocode Examples

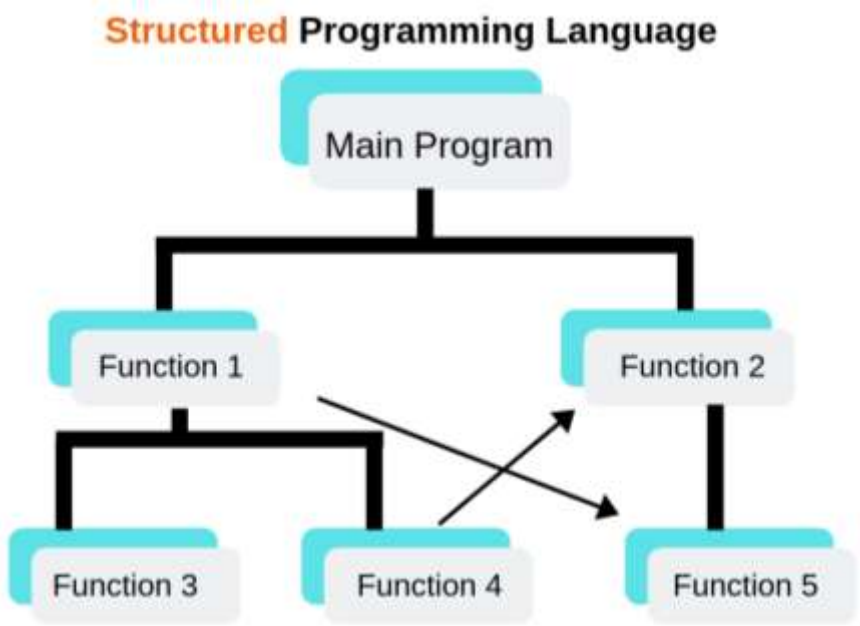
- Write a pseudocode to find the sum of series s=1+3+5+…. +N

```
1
2  begin
3      numeric nNum, nCtr, nSum
4      display "ENTER THE VALUE OF N : "
5      accept nNum
6      for(nCtr=1; nCtr<=nNum; nCtr=nCtr+2)
7      begin
8          nSum=nSum+nCtr
9      end
10     display "SUM OF SERIES : " nSum
11 end
12
```

Programming Paradigms

- The word “paradigm” means an example that serves as a pattern or a model. **Programming paradigms are the different patterns and models for writing a program.** The programming paradigms may differ in terms of the basic idea which relates to the program computation.
- Broadly, programming paradigms can be classified as follows:
  1. **Structured Programming,**
  2. **Object-Oriented Programming (OOP), and**
  3. **Aspect-Oriented Programming (AOP).** AOP is a new programming paradigm.

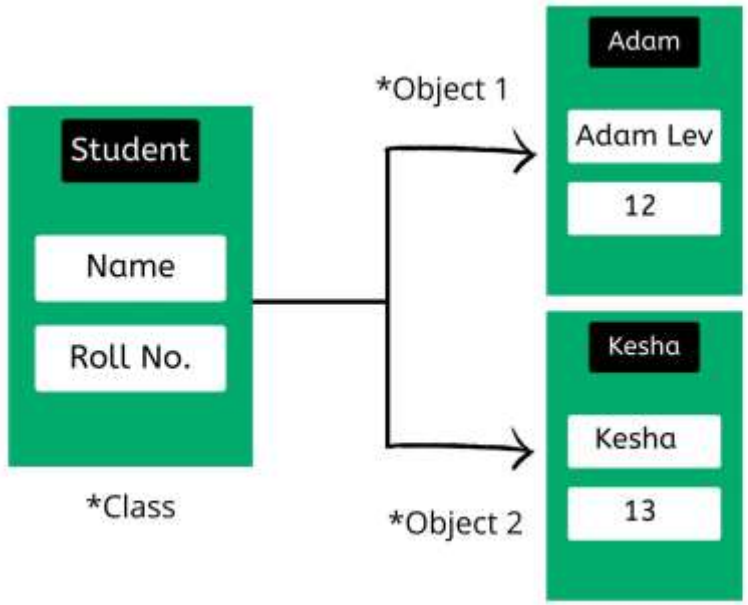
Programming Paradigms



Structured Programming:

- Structured programming is a programming paradigm aimed on improving the clarity, quality, and development time of a computer program **by making extensive use of subroutines, block structures and for and while loops.**
- Some examples of Structured programming are C, COBOL, Basic, PASCAL etc.

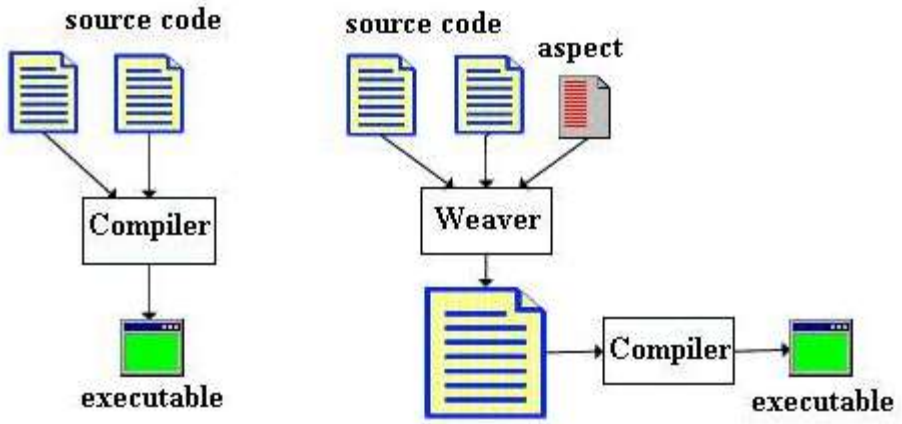
Programming Paradigms



Object-Oriented Programming (OOP):

- Object-oriented programming (OOP) is a programming paradigm that **represents concepts as "objects" that have data fields (attributes that describe the object) and associated procedures known as methods.**
- Objects, which are usually instances of classes, are used to interact with one another to design applications and computer programs.
- C++, Objective-C, Smalltalk, Java, C#, Perl, Python, Ruby and PHP are examples of object-oriented programming languages.

Programming Paradigms



Aspect-Oriented Programming (AOP):

- Aspect-oriented programming (**AOP**) is an approach to programming that allows global properties of a program to determine how it is compiled into an executable program.
- **AOP is a programming paradigm that aims to increase modularity by allowing the separation of cross-cutting concerns.** It will mainly help for cross-cutting programming concerns.
- Aspect-oriented programming entails breaking down program logic into distinct parts (so-called *concerns*, cohesive areas of functionality). **Typically, AOP techniques are used for logging, authorization, and authentication tasks.**

THE END