Comp 1400:

May 6th 2024

In class → Quiz / Test1 (May 29, 2024): 6%
→ Quiz / Test2 (June 12, 2024): 6%
→ Lab Assignments: 18%
- 5 lab assignment,
- 1 lab not graded.
- 2nd lab 2%
- rest are 4%

Pseudocode and Flowcharts

A typical programming task: can be divided into two phases:

* Problem solving phase and * Implementation phase

Definitions of Pseudocode and Flowchart

They are tools used for designing algorithms before actual coding.

Importance

→ Communication: easily discuss algorithms without the clutter of programming syntax

→ Clarification: break down complex ideas into simpler components

→ Error prevention: identify issues in logic before writing code.

Pseudocode

* It is a plain-language description of a computer program or algorithm, bridging human language and formal coding syntax.

Some Commonly used pseudocode writing styles

* Style 1: structured and verbose
* Style 2: simple and casual
* Style 3: Compact
* Style 4: line numbering. (this is the one we are going to use in this course)
* Style 5: Identification with line numbering.
* Style 6: Step-by-Step instruction

Guidelines

* Use CAPITAL letters for control structures (IF, ELSE, WHILE)
* Use meaningful names for variables and procedures
* Clearly specify the input and output.
* Avoid specific programming language syntax.

Flowchart

It is a visual representation of an algorithm or process.

Common Flowchart symbols

* Oval: start / End
* Parallelogram: input / output
* Rectangle: Process
* Diamond: decision (Yes / No)
* Arrow, Flow direction :

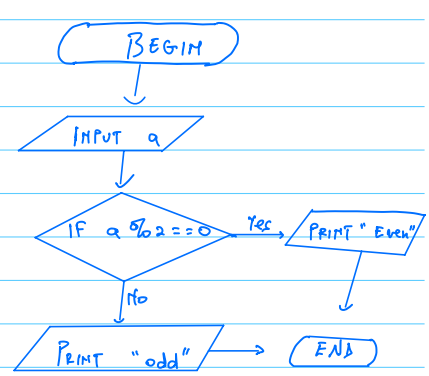
Exercise

→ Design an algorithm to check if a number is even or odd.

Pseudocode

```
BEGIN
INPUT a
IF  a % 2 == 0  THEN
    PRINT " The number is even"
ELSE
    PRINT " The number is odd"
END IF
END
```

Flowchart



Comparison: Pseudocode & Flowchart

* **Representation:** Pseudocode is a textual representation of an algorithm while a flowcharts provide a graphical representation.

* **Transition:** Often, a pseudocode is created first. Once the logic is solidified, it can be translated into a flowchart for visual understanding.

* **Clarity:** Pseudocode is more concise and to the point while flowcharts provide a clearer understanding for people who prefer visual learning.

* **Detail:** Pseudocode may contain more detailed steps, while flowcharts give a broad overview of the process.

When to use which?

-> Initial stages of development (Pseudocode)

-> Visualization (Flowcharts)

-> Documentation (Pseudocode)

Pitfalls to avoid.

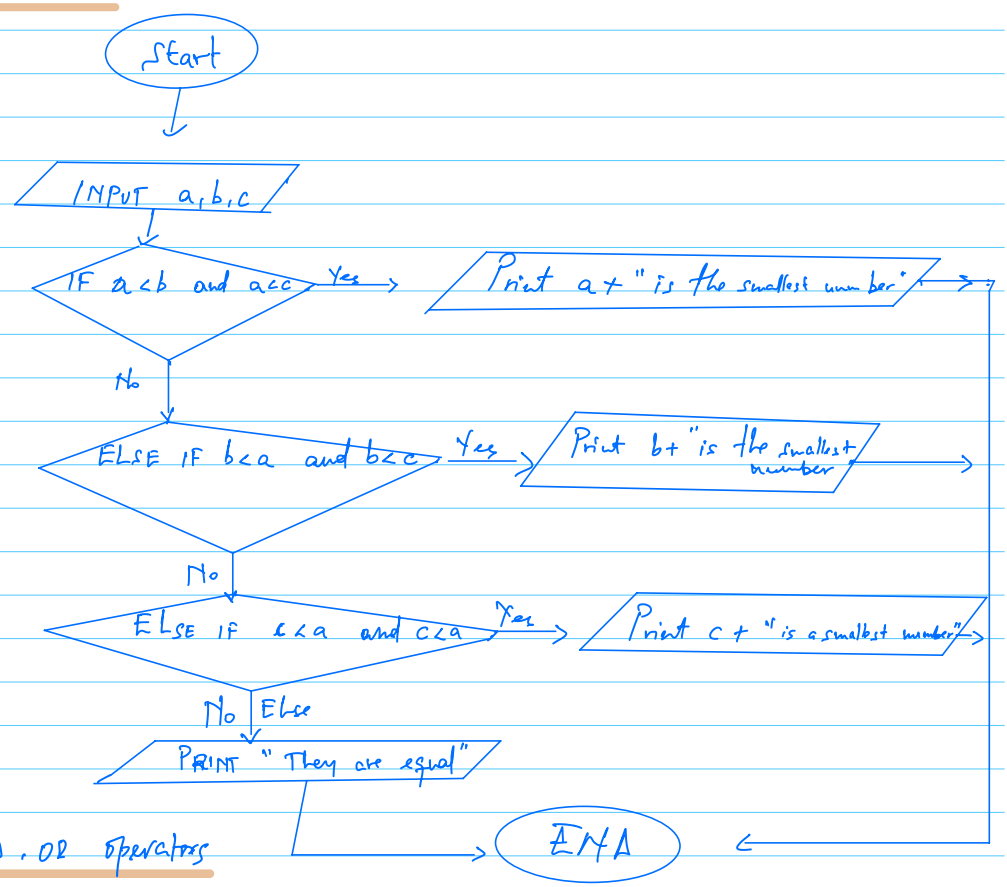
*> Overcomplication * Ambiguity (clear) * Ignoring standards

Exercise 2

Pseudocode

```
BEGIN
INPUT a, b, c
IF a<b && a<c THEN
    PRINT a + " is the smallest number"
  ELSE IF b<a && b<c Then
    PRINT b + " is the smallest number"
  ELSE IF c<a && c<b THEN
    PRINT c + "is the smallest"
  ELSE
    PRINT " They are all equal"
  END IF
END
```

Flowchart



AND, OR Operators

May 8th 2024

A	B	And (*)	OR (+)
T(1)	F(0)	F (0)	T(1)
F(0)	T(1)	F (0)	T(1)
T(1)	T(1)	T(1)	T(1)
F (0)	F (0)	F(0)	F(0)

is True as one (1)
False as zero (0)

Problem # 10: Leap Year Checker.

Check if a given year is a leap year or not. with a pseudocode and Flowchart

hint :

x A leap year is even divisible by 4

condition

If it is evenly divisible by 100 then its not a leap year.
unless it is evenly divisible by 400, then it is a leap year.

Redo Problem 9 !!!

```
If   year % 400 == 0 then
        PRINT "leap year"
Else IF  year % 100 == 0 then
    PRINT "Not a leap year"
Else IF  year % 4 == 0 then
        PRINT "leap year
Else
    PRINT "Not a leap year
END
```

IF year % 100 == 0 and

May 13th 2024

Loops

Types of loops

For loops: use when the number of repetition is known ahead of time

While loops: Use when the repetition is based on a condition, and you don't know how many times you need to loop

Do-While loops: Use when the repetition is based on a condition, but you need to execute the loop at least once

Exercises

① Design a pseudocode to compute the sum of the first N natural number. Pseudocode and flowchart.

Pseudocode.

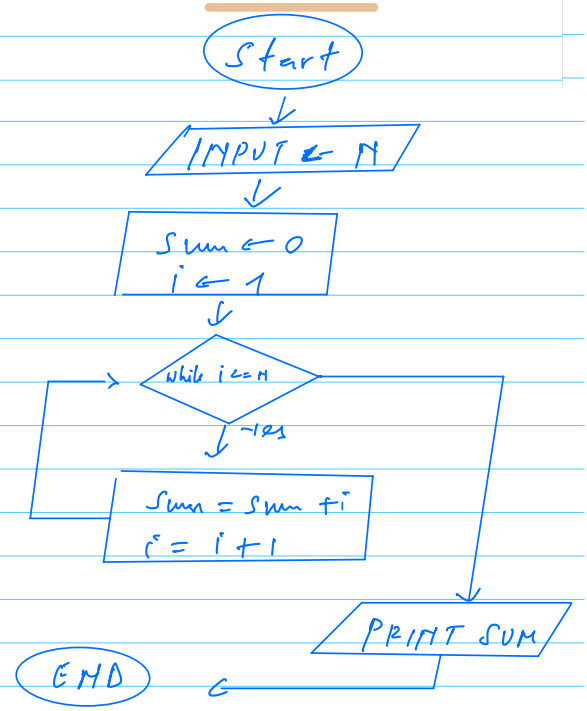
```
Start
INPUT: n

OUTPUT: sum
Set i ← 0, sum ← 0

  While i <= n

    Sum = sum + i
    i = i+1
  END LOOP
  PRINT SUM
  END
```

flowchart

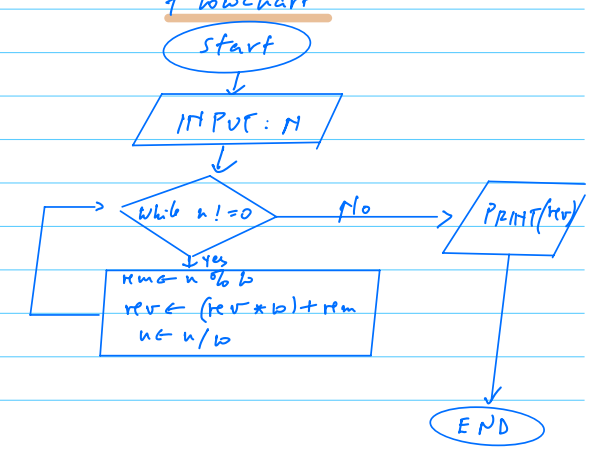


② Write an algorithm to reverse the digits of an number.

Pseudo code.

```
Start
INPUT: N
while (n!=0)
  set rem ← n % 10.
set rev ← (rev * 10) + rem;
    n = n/10 ;
END WHILE
PRINT (rev) ;  END
```

Flowchart



May 14th 2024

Introduction to C programming

// A First program in C

Ex
```
# include <stdio.h> // standard input/output header.
int main (void)
{
    printf ("Welcome to C! \n"); // printf ("f" stands for formatting)
} // end function main.
```

Begin with //, indicating that these two lines are comments.

You can also use /*-----*/

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. Produces a sound or visible alert without changing the current cursor position.
\\	Backslash. Insert a backslash character in a string.
\"	Double quote. Insert a double-quote character in a string.

Escape Sequences

Escape Sequences	Character
\b	Backspace
\f	Form feed
\n	Newline
\r	Return
\t	Horizontal tab
\v	Vertical tab
\\	Backslash
\'	Single quotation mark
\"	Double quotation mark
\?	Question mark
\0	Null character

* By using the escape sequence \" in a string to be output by printf, we indicate that printf should display a double quote.

eg: Printf(My folder is \" ss \\ test \"");

output My folder is "ss\test"

C program adding two integers

```
# include <stdio.h>
int main (void)
{
    int a, b, sum;
printf ("Enter first integer: \n");
scanf ("%d",&a);
printf ("Enter second integer \n");
scanf ("%d", &b);
sum = a+b;
printf("Sum is %d \n", sum);
}
```

* The %d in scanf stands for "decimal integer".
* The & is called address operator.

C operation	Arithmetic operator	Algebraic expression	C expression
Addition	+	$f + 7$	f + 7
Subtraction	-	$p - c$	p - c
Multiplication	*	bm	b * m
Division	/	x / y or $\frac{x}{y}$ or $x \div y$	x / y
Remainder	%	$r \bmod s$	r % s

Rules of Operator Precedence

Operator(s)	Operation(s)	Order of evaluation (precedence)
( )	Parentheses	Evaluated first. If the parentheses are nested, the expression in the *innermost* pair is evaluated first. If there are several pairs of parentheses "on the same level" (i.e., not nested), they're evaluated left to right.
* / %	Multiplication Division Remainder	Evaluated second. If there are several, they're evaluated left to right.
+ -	Addition Subtraction	Evaluated third. If there are several, they're evaluated left to right.
=	Assignment	Evaluated last.

Secure C Programming

* Avoid single-Argument printfs

eg printf ("Welcome to C! \n");

Should be written as:
puts ("Welcome to C!");

// We didn't include \n in the preceding string because puts adds it automatically.

* If you need to display a string without a terminating newline character use printf with two arguments.

eg: printf ("Welcome");

should be written as:

printf ("%s", "welcome");

→ these changes are responsible coding practices that eliminate certain security vulnerabilities.

Selection statements in C

* IF statement
* IF --- else statement.
* SWITCH selection (perform one of many different actions depending on the value of the expression)

IF --- else statement

```
int a = 15, b = 10
if (a = b) {
  print ("%d is equal to %d \n", a, b);
}
else {
  print ("%d is not equal to %d \n", a, b);
}
```

Conditional Operator

C provides the conditional operator (? :) which closely related to the IF ... else statement.

Syntax :

Condition ? Action for "true case : Action for "false" case.

eg : puts (grade >= 60 ? "Passed" : "Failed");

!!! Ask about slide 21 self assessment quiz !!!

Assignment Operators

(*) c = c + 3; is the same as c += 3;

Assignment operator	Sample expression	Explanation	Assigns
Assume: `int c = 3, d = 5, e = 4, f = 6, g = 12;`			
+=	c += 7	c = c + 7	10 to c
-=	d -= 4	d = d - 4	1 to d
*=	e *= 5	e = e * 5	20 to e
/=	f /= 3	f = f / 3	2 to f
%=	g %= 9	g = g % 9	3 to g

(*) c++; is the same as c += 1; or c = c + 1;

Operator	Sample expression	Explanation
++	++a	Increment a by 1, then use the new value of a in the expression in which a resides.
++	a++	Use the current value of a in the expression in which a resides, then increment a by 1.
--	--b	Decrement b by 1, then use the new value of b in the expression in which b resides.
--	b--	Use the current value of b in the expression in which b resides, then decrement b by 1.

```c
// Fig. 3.13: fig03_13.c
// Preincrementing and postincrementing.
#include <stdio.h>

// function main begins program execution
int main( void )
{
   int c; // define variable

   // demonstrate postincrement
   c = 5; // assign 5 to c
   printf( "%d\n", c ); // print 5
   printf( "%d\n", c++ ); // print 5 then postincrement
   printf( "%d\n\n", c ); // print 6

   // demonstrate preincrement
   c = 5; // assign 5 to c
   printf( "%d\n", c ); // print 5
   printf( "%d\n", ++c ); // preincrement then print 6
   printf( "%d\n", c ); // print 6
} // end function main
```

May 15th 2024

C Token

Keywords

Keywords in C Language

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
continue	for	signed	void
do	if	static	while
default	goto	sizeof	volatile
const	float	short	unsigned

Identifiers

e.g int x ; here x is the identifier which is the name to identify variables.

Constants

These are the values given to identifiers eg x = 10;
" Hello "

* Real constants is the same as floating point constants
* Single characters e.g: 'a', '1', 'm', 'F', etc
* String constants e.g: "good", "", "exi"

Terminal

* to create an output file in the terminal

gcc -o filename.c

* to execute the file in the terminal

./filename (// without .c)

* to edit the file in the terminal

vim filename.c

C program to sum two integers

```c

#include <stdio.h>

int main() {
    int a, b, sum;
    a= 10, b= 5;
    sum= a+b;
    printf("a = %d, b = %d\n Result = %d", a,b,sum);
    return 0;
}
```

// input from the user.

```c
//Program to sum two integers
#include <stdio.h>

int main() {
    int a, b, sum;
    printf("Enter two numbers: \n");
    scanf("%d %d", &a, &b);
    sum= a+b;
    printf("a = %d, b = %d\n Result = %d", a, b, sum);
    return 0;
}
```

May 21st 2024

Introduction to computers

* **Hardwares** refer to the various devices that consist a computer
  e.g: keyboard, screen, mouse, hard disk, e.t.c)

* **Softwares:** refer to the programs that run on a computer

Logical units

* **Input unit:** this is the receiving section that obtains information from input devices and places it at the disposal of the other units for processing