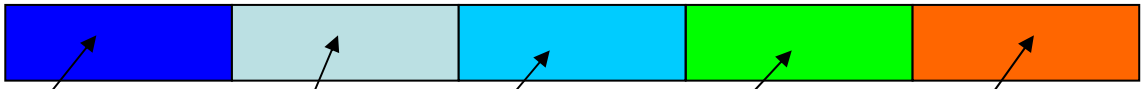
Advanced Mechatronics Design

Software Tools

Processors and Processing

- Microcontrollers & Software Tools
  - Programming Model
  - Languages
  - Applications
  - C programming
  - LabVIEW
  - Matlab / Simulink

Programming Model
Generic Instruction Form



Instruction
Operand 1, or Address of it
Operand 2
Where to store results
Where is new instruction

If we have 131 Instructions and Address Space of 64K, what would be the Instruction length?

Programming Model with Registers

Instruction
Operand 1
Operand 2
Where to store results
Where is new instruction

ID (Instruction Decoder)
ACC (Accumulator)
Mem (Memory Location)

ACC (Accumulator)
PC (Program Counter,
Or Instruction Pointer-IP)
PC=PC+1…

Languages

- Binary: 0110 0001 1111 1010 1100 ***GEN 1***
- ASM: ADD data1 ***GEN 2***
- High Level Languages: ***GEN 3***
  - Modular: Fortran, C

    Data

    Functions
  - Object Oriented: C++, JAVA

    Data & ***Functions*** → ***Objects***

Languages

- Database Programming: ***GEN 4***
  - MS Access, Oracle
  - No Programming, i.e. just specify what is requested from the results (conditions to be fulfilled)
- Graphical Programming
  - LabVIEW,
  - Simulink / Matlab
  - Robotino®View

Introduction, C Environment

ANSI C, standardised in 1989 (ANSI, ISO)
High Level Language, Structured Programming
C Development Environment for a UNIX based system

Editor	*myfile.c is created and stored on the disk*
Preprocessor	*Preprocessor program process the code*
Compiler	*Compiler creates object code and stores it on the disk*
cc myfile.c,	***gcc myfile.c a.out***
Linker	*Linker links the object code with libraries, creates* ***a.out*** *and stores it on the disk*
Loader	*Loader loads program in memory*
CPU	*Program execution*

Making an .exe file

- Source file name.c + stdio.h, #include files

Compiler

- name.obj + Library files + Other user files

Linker

- Machine language file, i.e. executable file, or binary file ***01100110100010101000111***

C Building Blocks

Variables, Input/Output, Operators, Comments,

```
/*      A simple program in C */
//      comment

        #include <stdio.h>  /* include information about Standard Library*/
        main()              /*define a function called main*/
                            /*that receives no argument values*/
        {                   //statements of main are enclosed in //braces {}

           printf("Hello World\n");
                            /*main calls library function printf to print this
                            sequence of characters;     \n represents the new line
                            character */
        }
```

Variable types

int	**INTEGER**	**int sum;** **sum = 55;** **int event = 10;** **int int1, int2, sum2;**

Whole numbers, both positive and negative *16 or 32 bits wide*

Unsigned integers, positive values only

Short and Long integers **short, long**

char	**CHARACTER**	**char letter;** *8 bits (-128 to 127)* **letter = 'E';** *0 to 255*

Variable Types in Memory

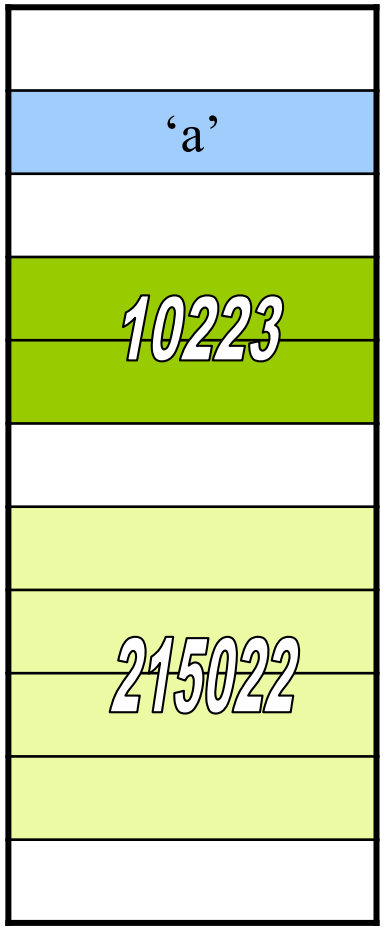
-128 to 127

-32768 to 32767

-2,147,483,648

to

2,147,483,647

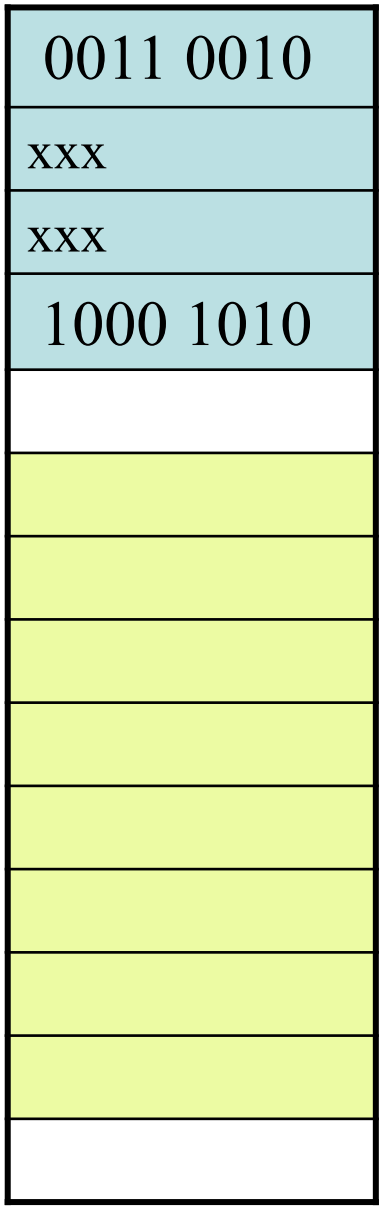


Character (type **char**)

Integer (type **int**)

Long integer (type **long**)

Variable Types in Memory



Floating point
(type ***float***)

10^{-38} to 10^{+38}
7 digits precision

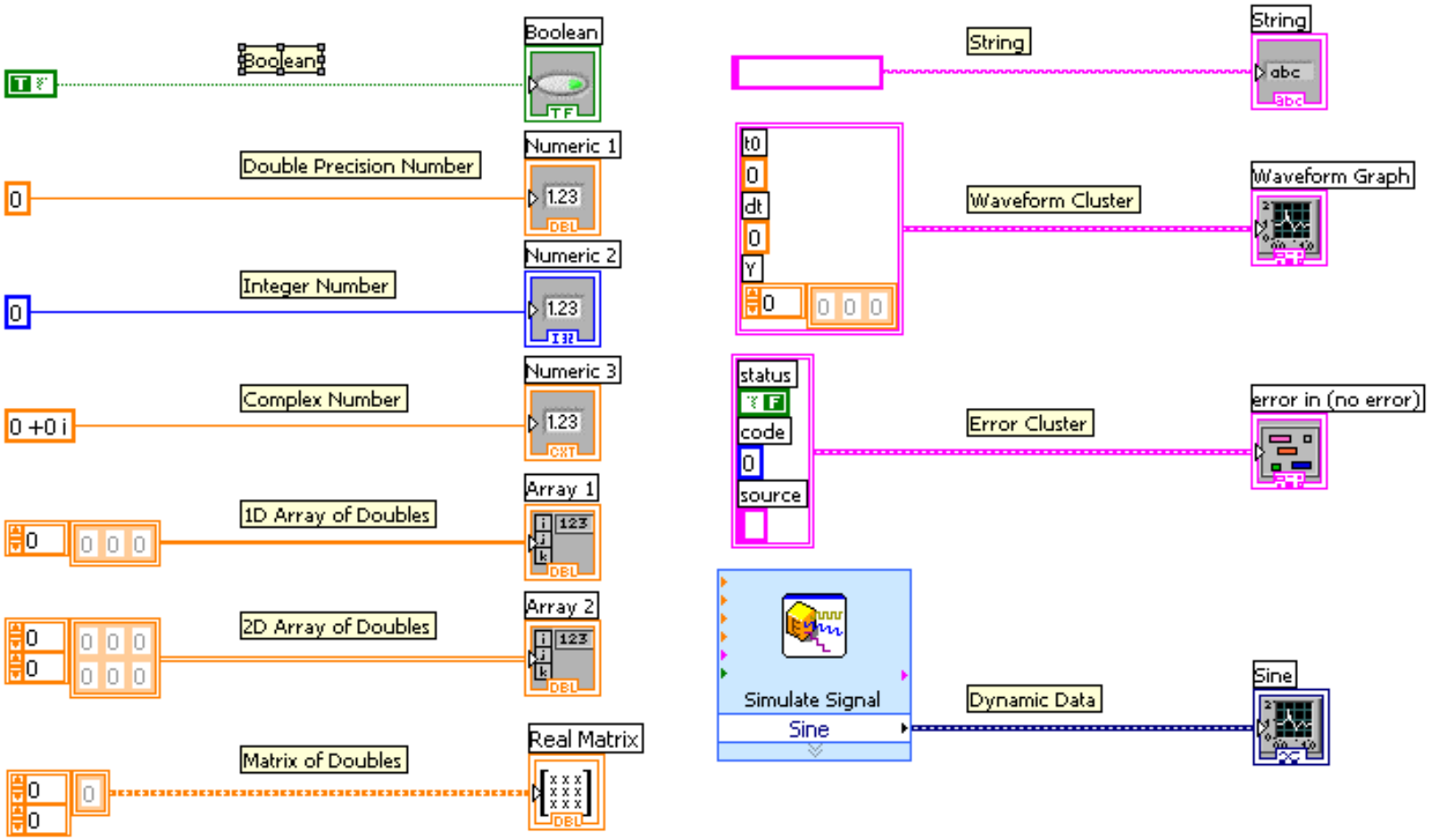
Double precision
Floating point
(type ***double***)

10^{-308} to 10^{+308}
15 digits precision

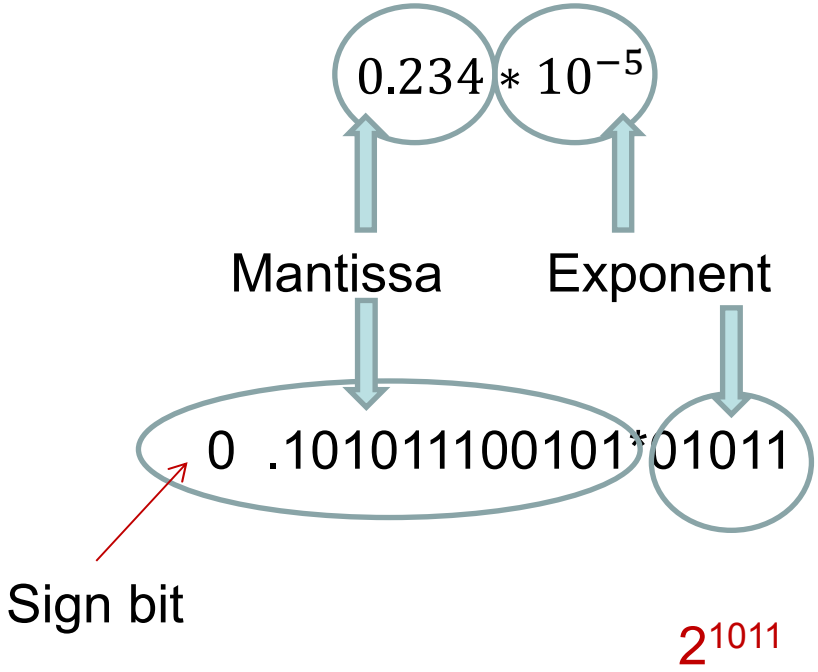
There is also
type ***long double***
with
10 bytes

10^{-4932} to 10^{+4932}
19 digits precision

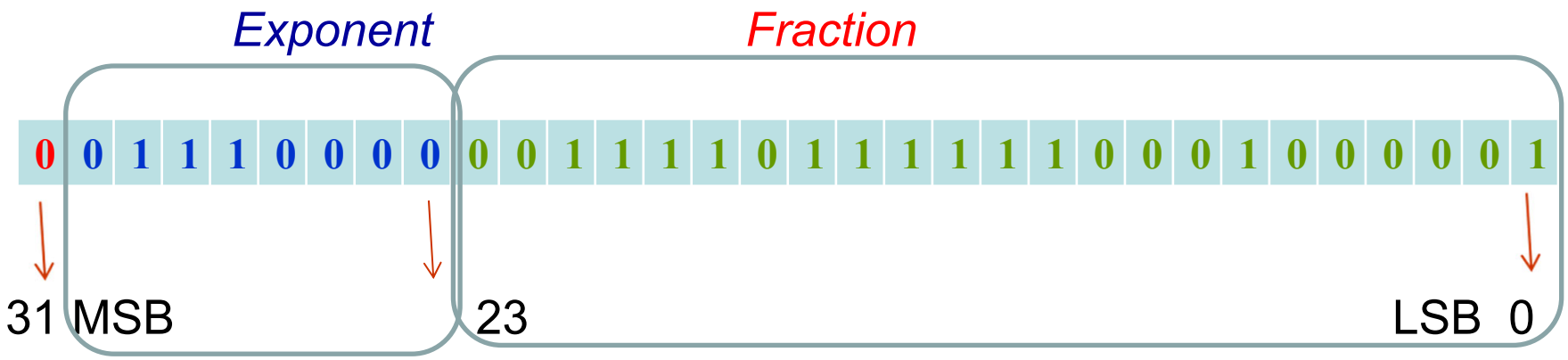
Data Types Found in LabVIEW



Data Representation



IEEE Binary 32 bit Floating Point



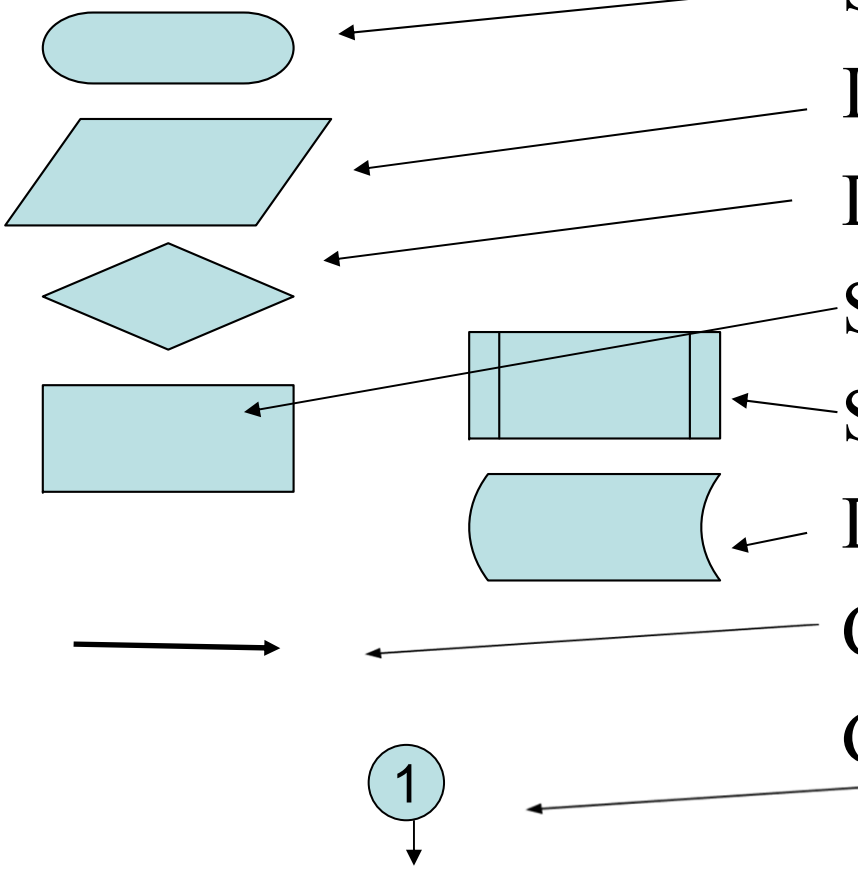
Sign	*bit 31*	
Exponent	*bits 23-30*	*8 bits*
Fraction	*bits 0-22*	*23 bits*

MSB Most Significant Bit
LSB Least Significant Bit

Variables – Modified Types

short int	usually 16 bits
long int	usually 32 bits
unsigned int	16 / 32 bits just positive numbers
unsigned char	0-255

Flowchart Symbols



Start / End

Input / Output Data

Decision Block

Statements Block, Process { }

Subroutine

Data from the Storage

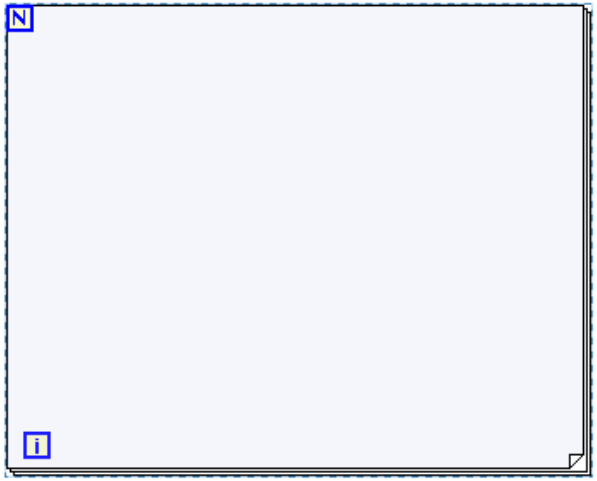
Connector

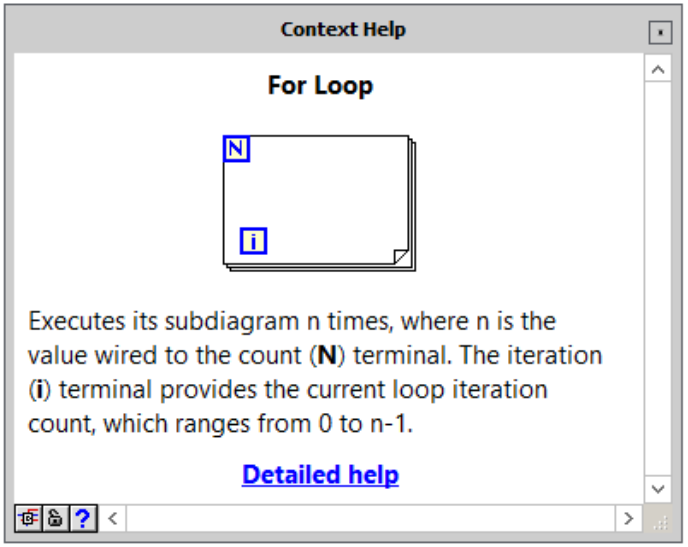
Go to the next page

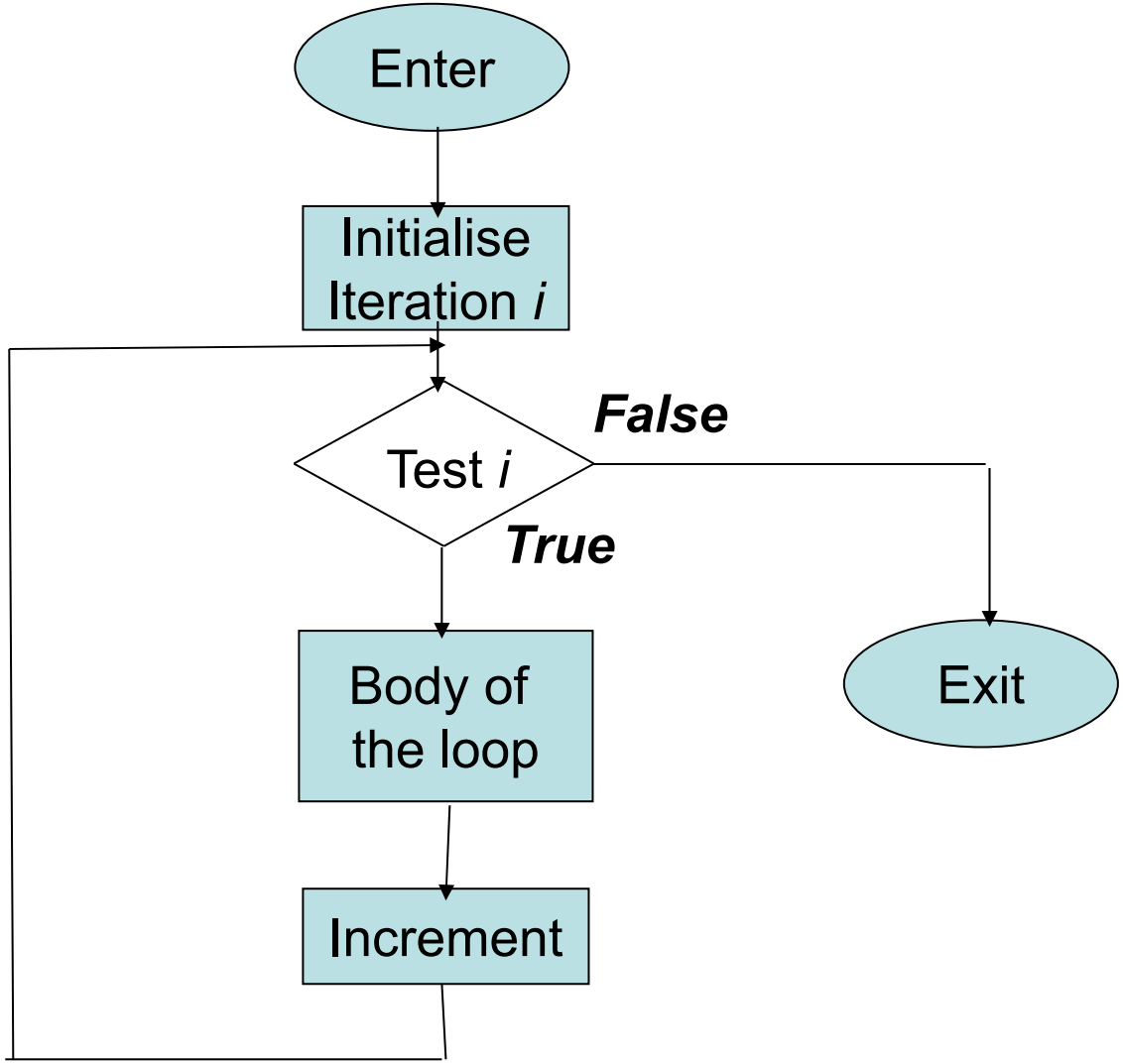
Loops

- The ***for*** Loop, useful when we know the number of repetitions
- The ***while*** Loop, Testing conditions first and performing the instruction (s) after. May not go through the loop at all.
- The ***do while*** Loop. Goes through at least once.

For Loop


N
i


Context Help
For Loop
N
i
Executes its subdiagram n times, where n is the value wired to the count (N) terminal. The iteration (i) terminal provides the current loop iteration count, which ranges from 0 to n-1.
Detailed help


Enter
Initialise Iteration *i*
Test *i*
False
True
Body of the loop
Exit
Increment

For Loop – Please Explain

```
void main(void)
{
    int count;
    int total;
    for (count=0, total=0; count<10; count++)
        {
         total+=count;
         printf("\ncount=%d,total=%d",count,total);
         }
}
```

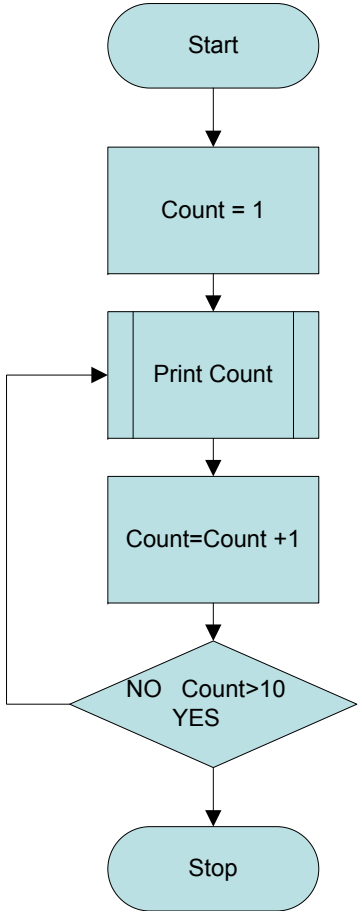
Loop / Repetition Examples

Write a program in C that print integer numbers from 1 to 10.

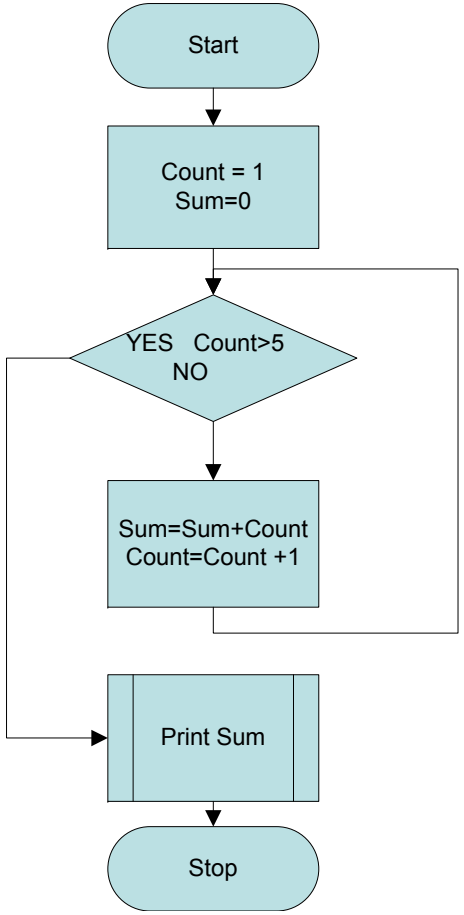
Write a program in C that will add integer numbers from 1 to 5 and print the sum and average number.

- Design flowchart
- Implement the flowchart in software

Flowchart



Flowchart



Add steps to Calculate and Print average

Using **for** Loop

```
#include <stdio.h>
main( )
{
    int  count;
    for( count = 1; count <= 10; count++ )
        printf("%d ", count );
    printf("\n");
}
```

While Loops

while **while (expr) stmt**

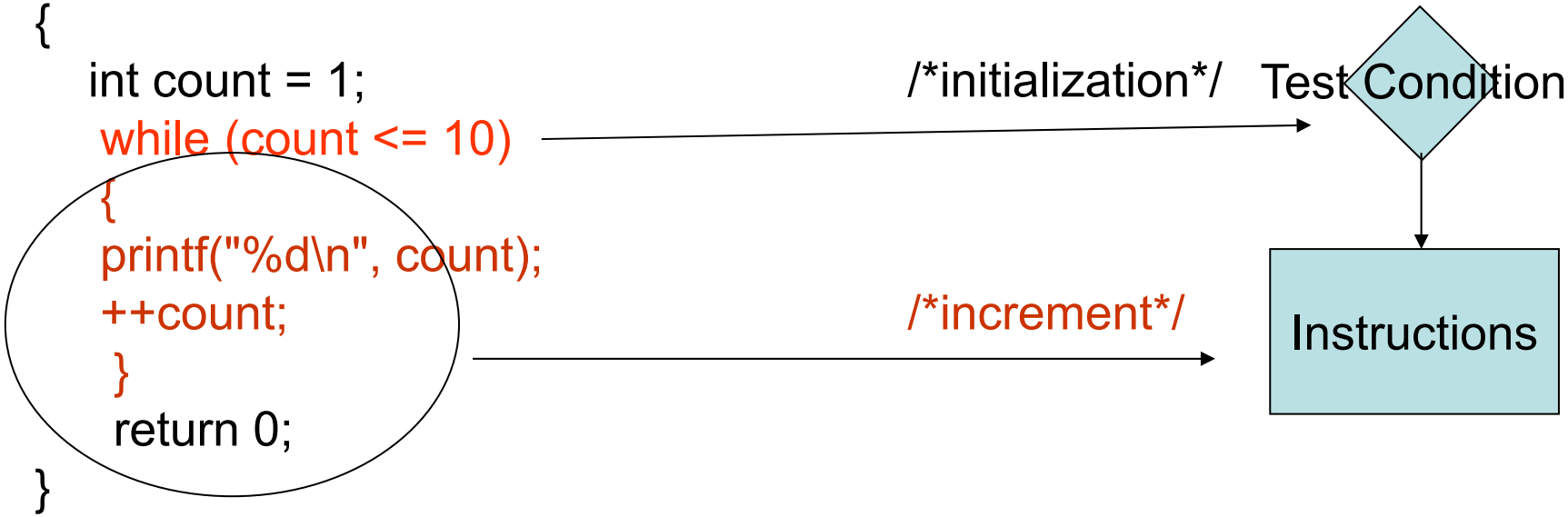
continue executing `stmt' while `expr' evaluates to non-zero

do **do** stmt **while**(expr)

do `stmt' at least once, then while `expr' evaluates to non-zero

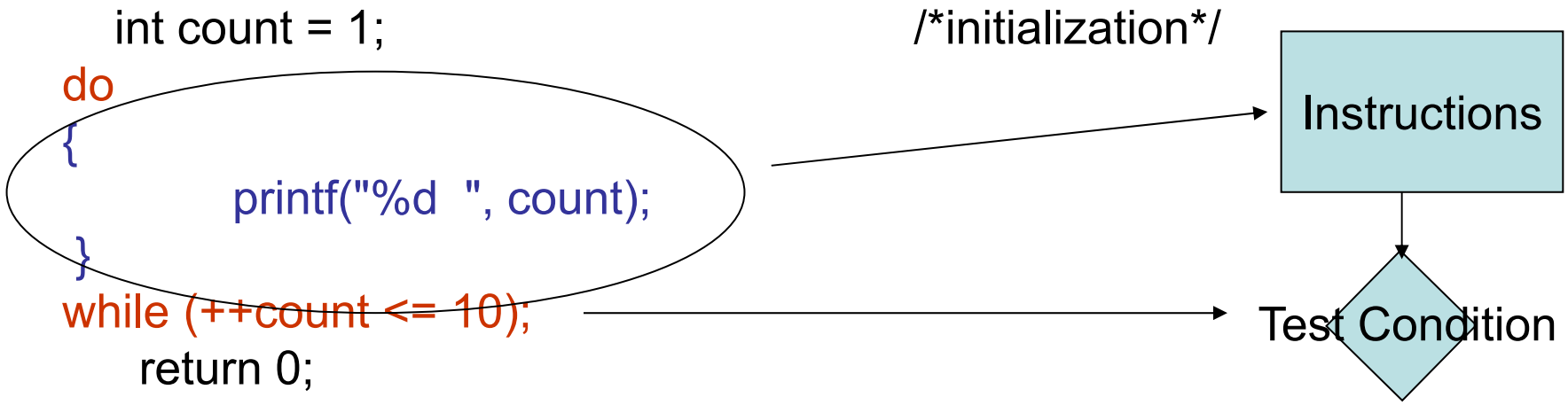
Counter controlled repetition, using **while**

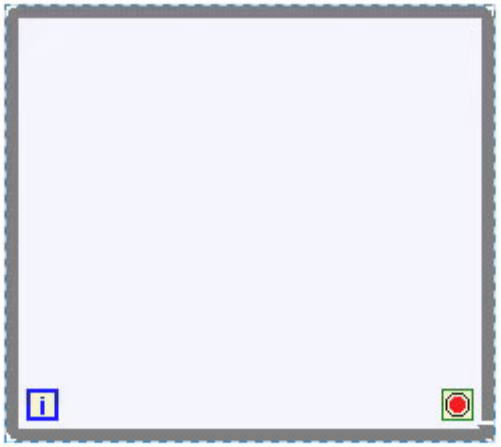
```
#include <stdio.h>
main()
{
    int count = 1;
    while (count <= 10)
    {
    printf("%d\n", count);
    ++count;
    }
    return 0;
}
```

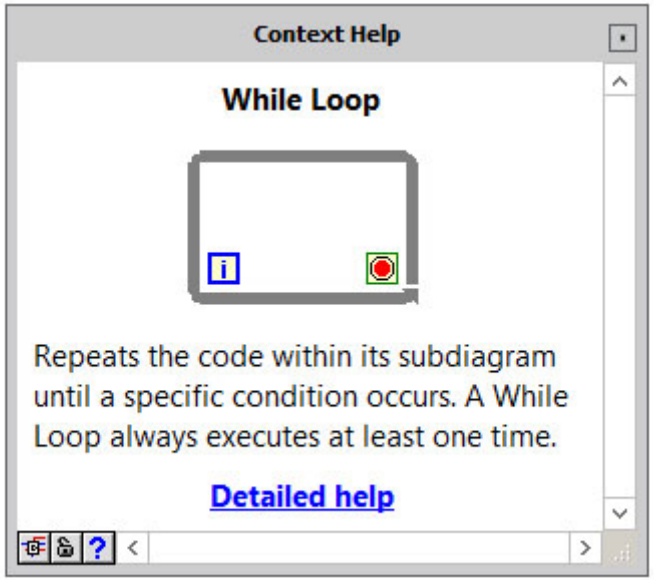


Counter controlled repetition, using **do/while**

```
#include <stdio.h>
main()
{
   int count = 1;                    /*initialization*/
do
{
          printf("%d  ", count);
}
while (++count <= 10);
     return 0;
}
```






Context Help
While Loop
Repeats the code within its subdiagram until a specific condition occurs. A While Loop always executes at least one time.
Detailed help

Average Number

Draw flowchart for the program that will ask (prompt) user to input an integer number,

$$n > 0,$$

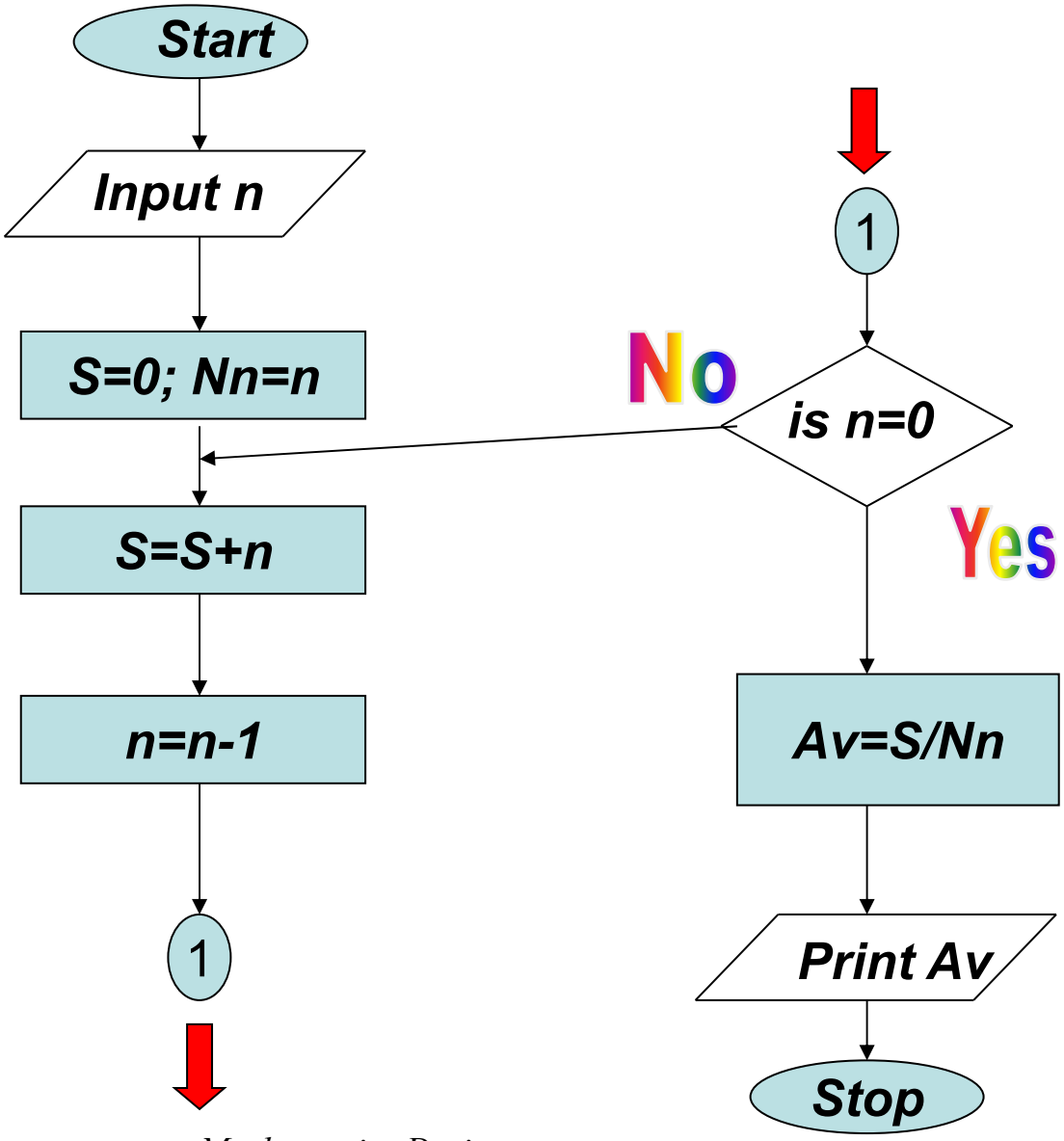
calculate average of all numbers from

$$1 \text{ to } n$$

and print result.

Average Number

S = Sum
Av = Average

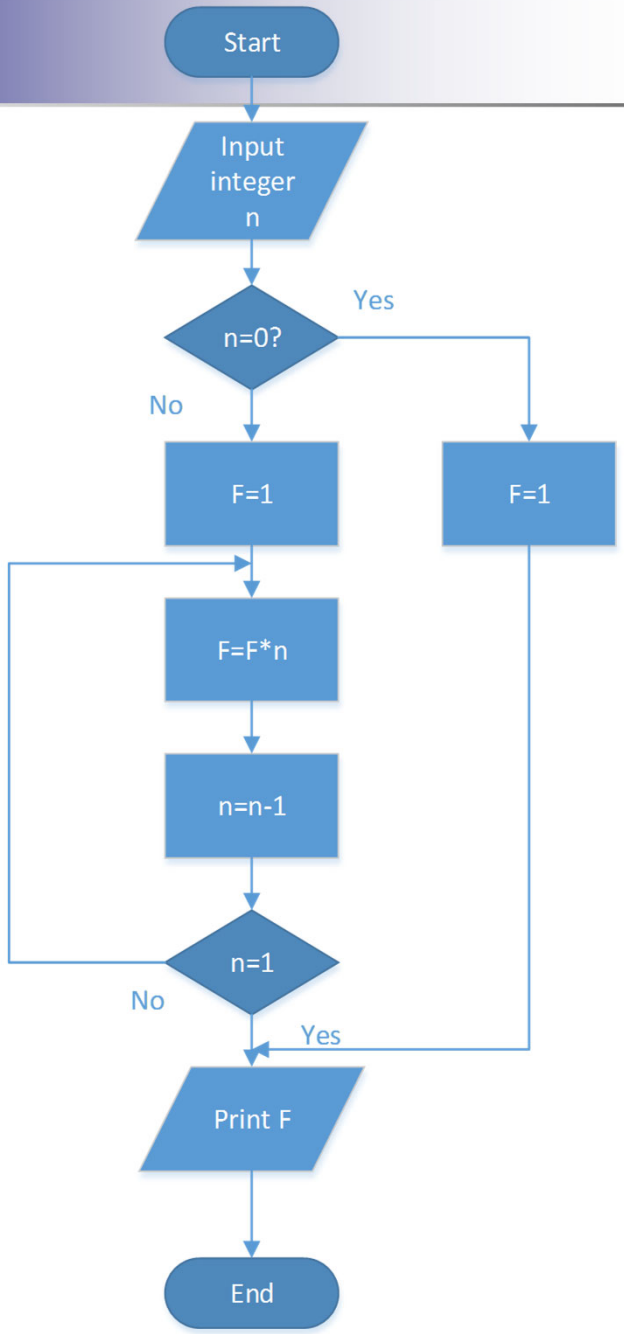


Flowcharting and Programming

Write a program in C that will prompt used to input an integer number, n, calculate function n! (Factorial) and print result.

$$n! = \begin{cases} 1, for\ n = 0 \\ \prod_{k=1}^{n} k\ , for\ n \geq 1 \end{cases}$$

- Design flowchart
- Implement the flowchart in software


Start
Input integer n
n=0?
Yes
No
F=1
F=1
F=F*n
n=n-1
n=1
No
Yes
Print F
End

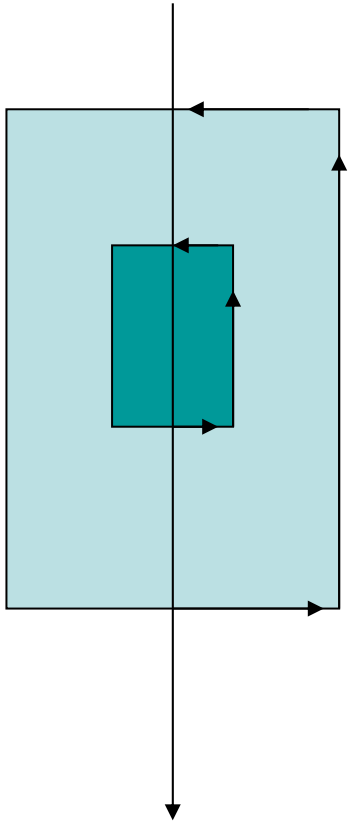
Combinations

- Design an algorithm i.e. draw a flowchart for the program that should calculate the number of k-combinations from a given set S of n elements.

Combinations

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

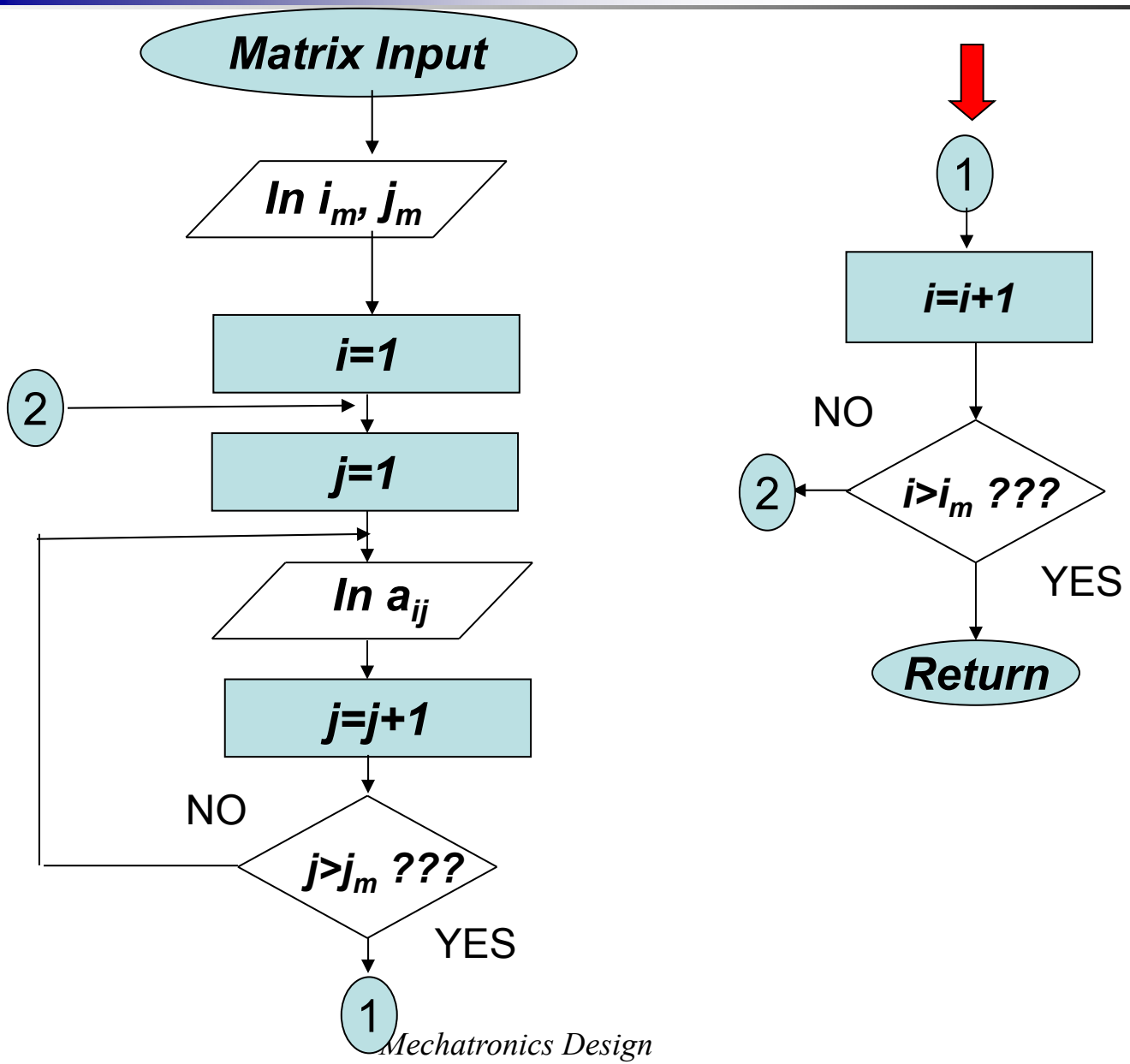
Nested Loops



Nested Loops

- Draw a flowchart for the program that will first
  - prompt user to input Matrix A dimensions i_m and j_m and then
  - input the elements of the matrix A, a_{ij}, i=from *1* to i_m, j=[1 to j_m]

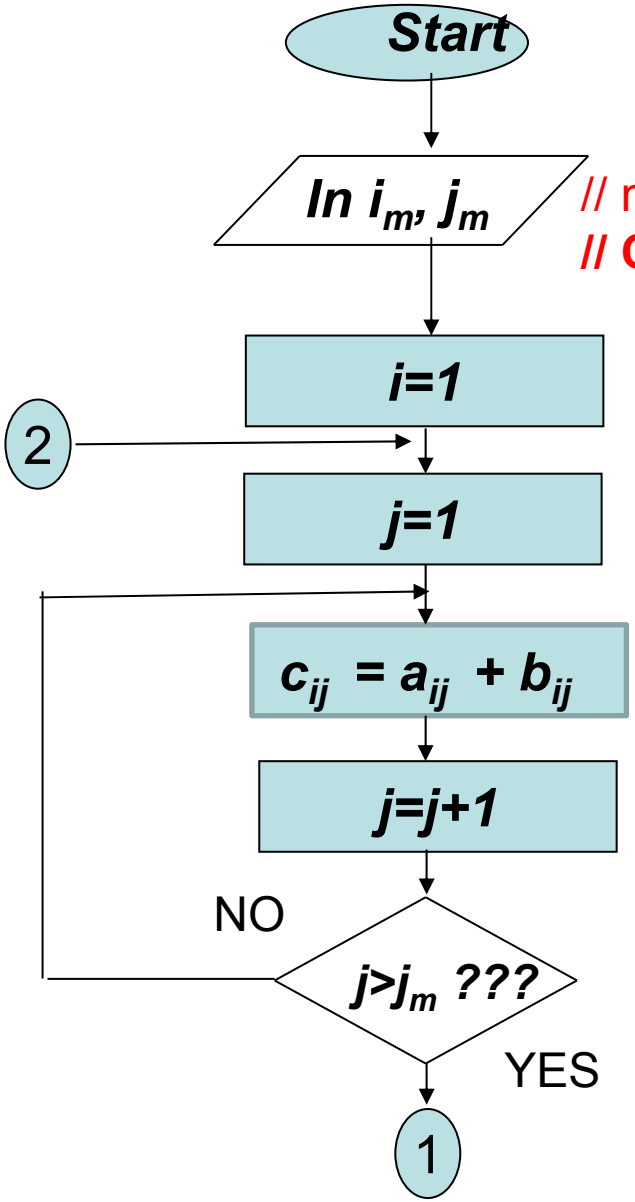
$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$


Matrix Input
In i_m, j_m
i=1
2
j=1
In a_{ij}
j=j+1
NO
$j>j_m$???
YES
1
1
i=i+1
NO
2
$i>i_m$???
YES
Return

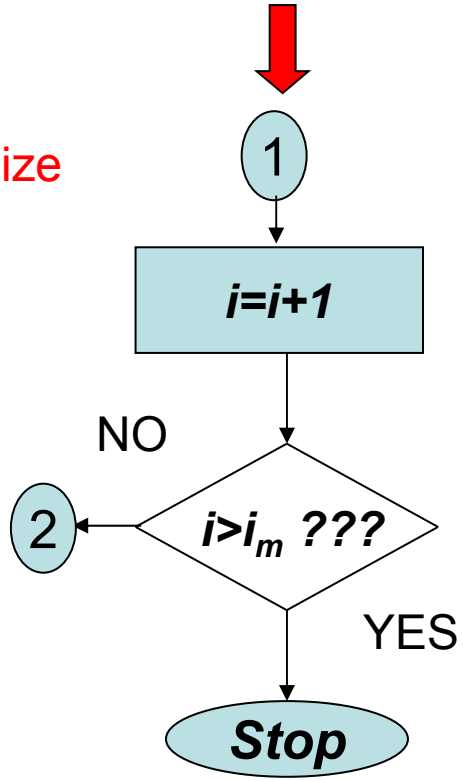
Exercises

- Draw a flowchart for the program that will add two matrices **C=A+B.** Data is already in the memory.
- Matrices **A, B** and **C** have the same size.

C=A+B



// matrices must be the same size
// C=A+B



$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} + \begin{pmatrix} 2 & 4 & 6 \\ 5 & 10 & 11 \\ 9 & 16 & 18 \end{pmatrix} = \begin{pmatrix} 2 & 4 & 6 \\ 5 & 10 & 11 \\ 9 & 16 & 18 \end{pmatrix}$$

Exercises

- Draw a flowchart for the program that will multiply two matrices **C=A*B.** Data is already in the memory.

A is $i_m * k_m$

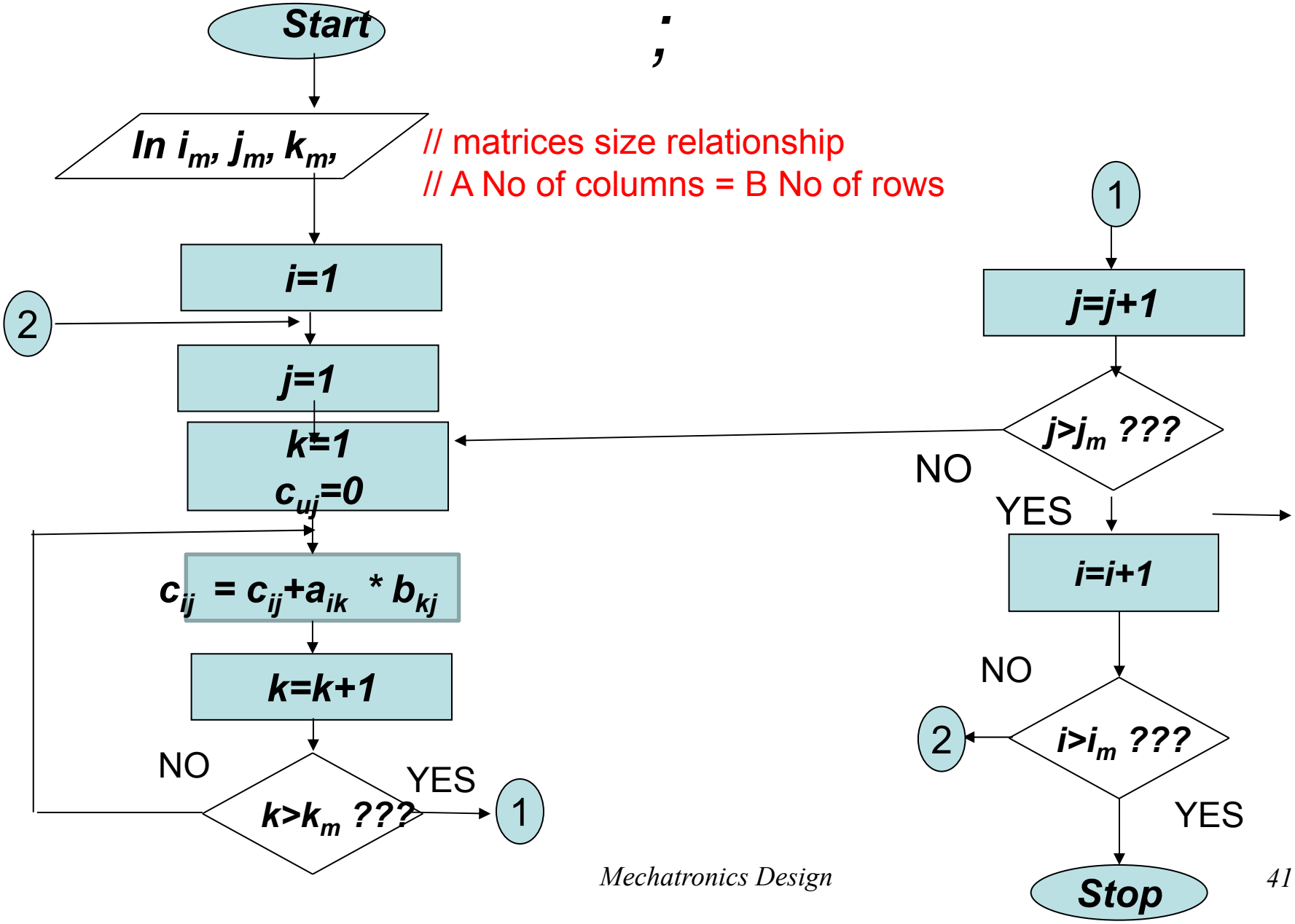
B is $k_m * j_m$

C is $i_m * j_m$

$$c_{ij} = \sum_{k=1}^{k_m} a_{ik} * b_{kj}$$

C=A*B

;


Start
In i_m, j_m, k_m,
// matrices size relationship
// A No of columns = B No of rows
i=1
2
j=1
k=1
c_{uj}=0
$c_{ij} = c_{ij}+a_{ik} * b_{kj}$
k=k+1
NO
$k>k_m$???
YES
1
1
j=j+1
$j>j_m$???
NO
YES
i=i+1
NO
2
$i>i_m$???
YES
Stop

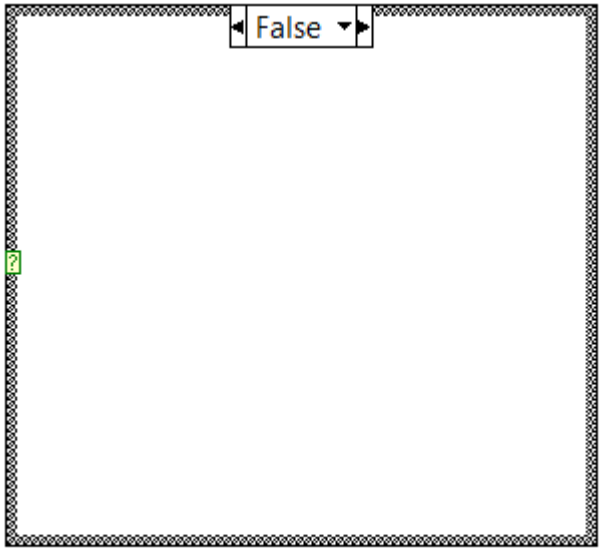
C=A*B

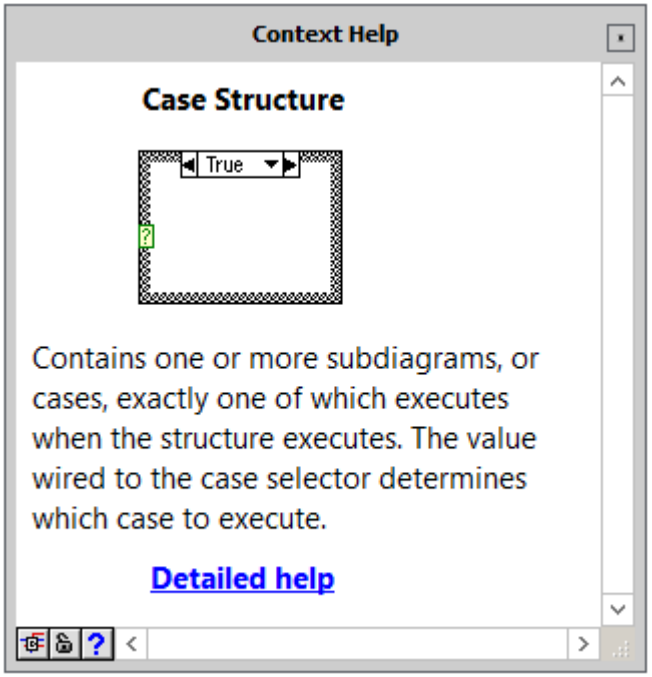
$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} * \begin{pmatrix} 1 & 2 \\ 1 & 5 \\ 2 & 8 \end{pmatrix} = \begin{pmatrix} 9 & 36 \\ 21 & 81 \end{pmatrix}$$

The Switch Statement Or Case Structure

```
switch    (op) // "switch variable" integer or character variable (or expression)
    {
    case 'a':                //if switch variable ='a'
        statement;
        statement;
        break;
    case 'b':                //if switch variable ='b'
        statement;
        statement;
        break;
    default:
        statement;
        statement;
    }
```

/* case labels must be numbers or char literals;
when a case or default is chosen, execution continues until
break or end of block */


False


Context Help
Case Structure
True
Contains one or more subdiagrams, or cases, exactly one of which executes when the structure executes. The value wired to the case selector determines which case to execute.
Detailed help

Thank you,
Questions