CS F111: Computer Programming

(Second Semester 2020-21)

Lect 11: Preprocessors and if..

Dr. Nikumani Choudhury
Asst. Prof., Dept. of Computer Sc. & Information Systems
nikumani@hyderabad.bits-pilani.ac.in

BITS Pilani

Hyderabad Campus

Character i/o using getchar() and putchar()

```
char c;
...
c = getchar ();
...
```

```
char c = 'A';
...
putchar (c);
...
```

getchar () accepts any character keyed in including RETURN and TAB

What would be printed here:
putchar ('\n');

```
int main(void)
{
  int c;
  while((c = getchar())
                != EOF )
putchar(toupper(c));
fflush(stdout);
return 0;
}
```

```
#include <stdio.h>
main()
{
   char line[50];
   gets (line);
   puts (line);
}
```

gets vs scanf:

- reads a string, reads any data type.
- Stops at NL/EOF, stops at WS/NL/EOF.

Preprocessors

#include – includes the content of an alternate file

#define – defines a macro

Macros Substitution

Definition has the form :

```
#define name replacement_text
```

- **Anywhere name occurs it is replaced by the replacement_text.**
- **This definition of macro can be anywhere it can be inside main() outside main().**
- **Name does not have any data-type associated with it.**
- **Macro definitions are not variables and cannot be changed by your program code like variable**

Example

- **Here after preprocessing PI is replaced by 3.146 everywhere it occurs**

```
#include<stdio.h>
#define PI 3.1416

int main()
{
    double areaOfCircle, perimeter;
    double r;

    scanf("%lf", &r);

    areaOfCircle = PI*r*r;
    perimeter = 2*PI*r;

    printf("area = %lf perimeter = %lf", areaOfCircle, perimeter);
    return 0;
}
```

Example

```
#define MAX 500
int a = MAX;

#define AGE (20 / 2)

#define int abc
abc x;
```

Macros with Argument

- Macros can have argument too.
- Will generate different replacement text for different calls of the macros.
- There can be any number of arguments.
- Actual argument of macros are not evaluated since it's only replacement

Macros with Argument

```
#define SQUARE(x) x*x


int v1 = SQUARE(4);
int v2 = 64/SQUARE(4);
int v3 = SQUARE(2+3);
```

Example

```c
#include<stdio.h>
#define SQUARE(x) ((x)*(x))  // macro with argument

int main()
{
    int val1 = SQUARE(4);   // replaced text is ((4)*(4))
    int val2 = 64/SQUARE(4); // replaced text is 64/((4)*(4))
    int val3 = SQUARE(2+3); // replaced text is ((2+3)*(2+3))

    printf("%d %d %d\n", val1, val2, val3); // 16 4 25

    return 0;
}
```

Control Structures

- All programs written in terms of 3 control structures
  - Sequence structures: Programs executed one after the other in the order written.
  - Selection structures: C has three types: if, if…else, and switch.
  - Repetition structures: C has three types: while, do…while and for.

Control Statements

- **Decision / Branching:**

  Allow different sets of instructions to be executed depending on the outcome of a logical test

  Whether TRUE (non-zero) or FALSE (zero)
  - **if construct statement**
  - **switch case statement**
  - **ternary operator statement**

- **Looping:**

  Some applications may also require that a set of instructions be executed repeatedly, possibly again based on some condition.
  - **while**
  - **for**

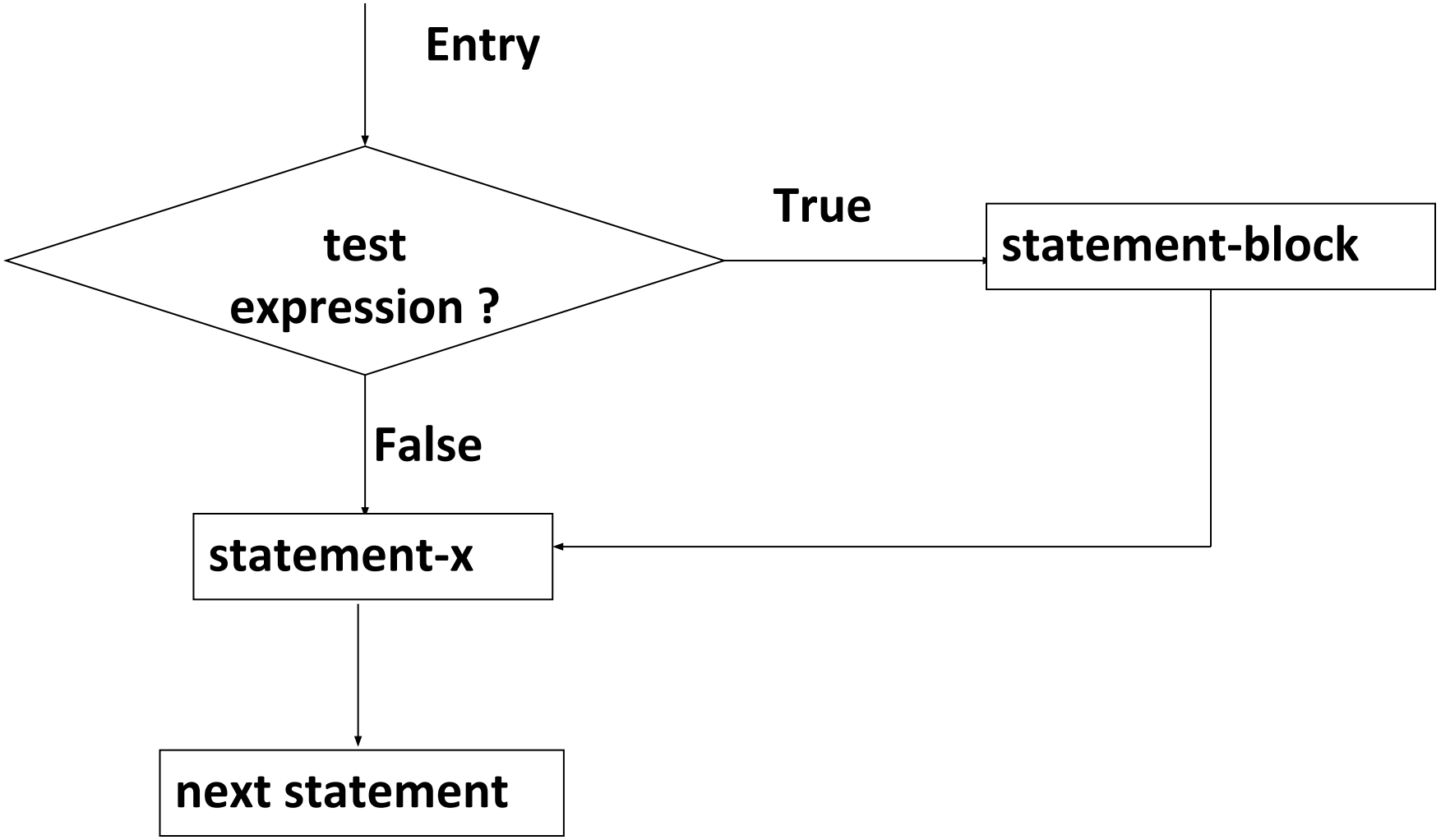
if Statement

```
if (expression)
      statement;
statement-x;
```

```
if (expression)
{
  set of statements;
}
statement-x;
```

- first expression is evaluated
- If expression is non-zero then the statement is executed else not

Flowchart of if control



if Statement

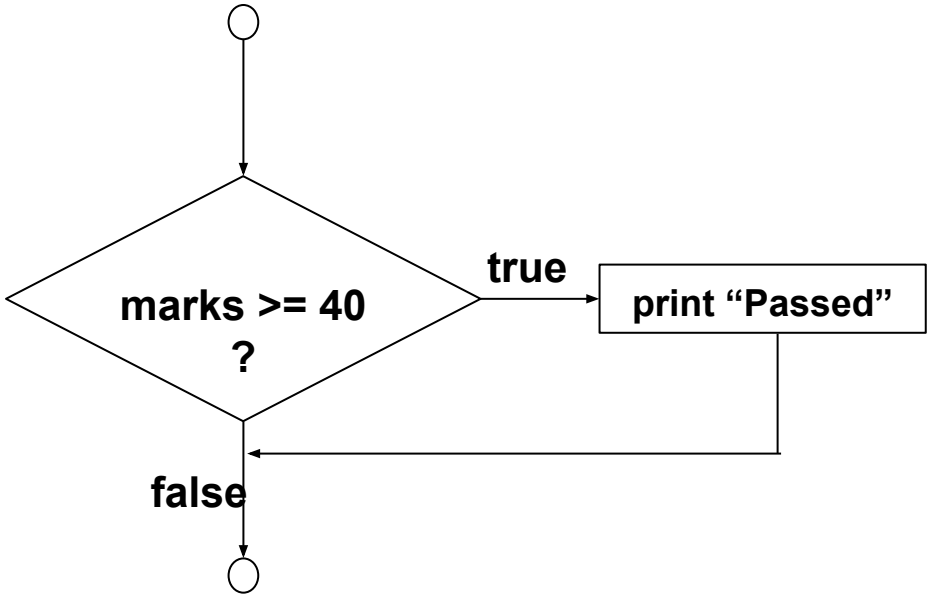
Caution : **if(expr); if(expr)**

{

}

Decision making with if statement

```
if ( marks >= 40 )
 printf( "Passed\n" );
```



A decision can be made on any expression.
zero - false
nonzero - true

Evaluation of expression

this expression	is true if
x == y	x is equal to y
x != y	x is not equal to y
x < y	x is less than y
x > y	x is greater than y
x <= y	x is less than or equal to y
x >= y	x is greater than or equal to y

```
if ( 3 + 2 % 5 )
    printf ( "This works" ) ;
if ( -5 )
    printf ( "Surprisingly even this works" ) ;
```

Recall that in C a non-zero value is considered to be true, whereas a 0 is considered to be false.