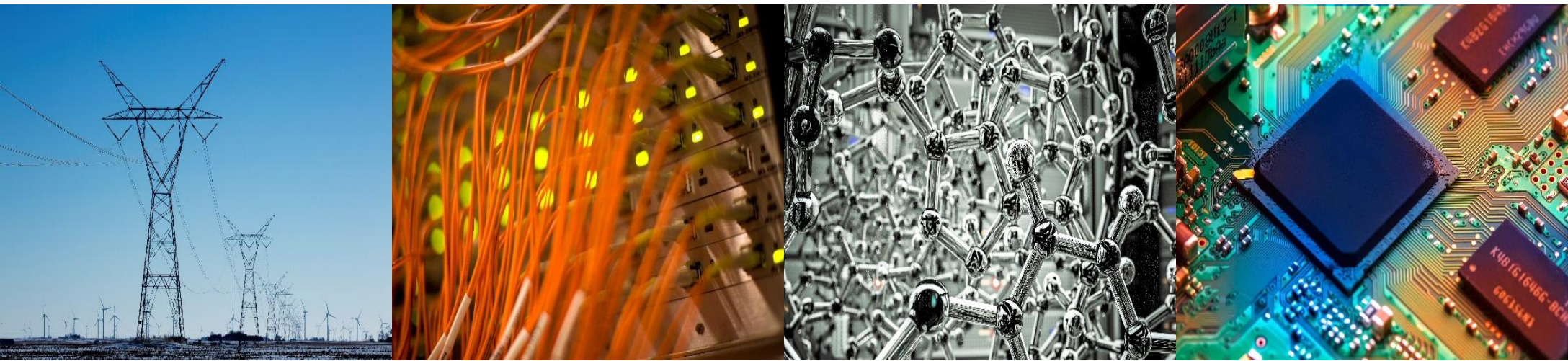# ECE 220 Computer Systems & Programming

## Lecture 6 – Introduction to C: Control Structures
## September 12, 2024



- **Quiz1 is next Monday through Wednesday @ CBTF**

![I ILLINOIS Electrical & Computer Engineering GRAINGER COLLEGE OF ENGINEERING]

# Lecture 5 Review

❖ **global vs. local (scope)**

```
int global_x = 5;
int main(){
   int local_x = 10;
   printf("global=%d, local=%d\n", global_x, local_x);
   {
      int local_x = 15;
      global_x = 20;
   }
   printf("global=%d, local=%d\n", global_x, local_x);
   return 0;
}
```

❖ **const vs. static (qualifier)**

```
const int x = 0;
static int y = 0;
```

❖ **' = ' vs. ' == '**

```
int x = 1, y = 2;
int result = (x == y);
```

❖ **pre vs. post decrement**

```
int x = 5, y = --x;
int x = 5, y = x--;
```

ECE ILLINOIS

# Control Structures

Conditional Constructs

- if

- if - else

- switch

Iteration Constructs (loops)

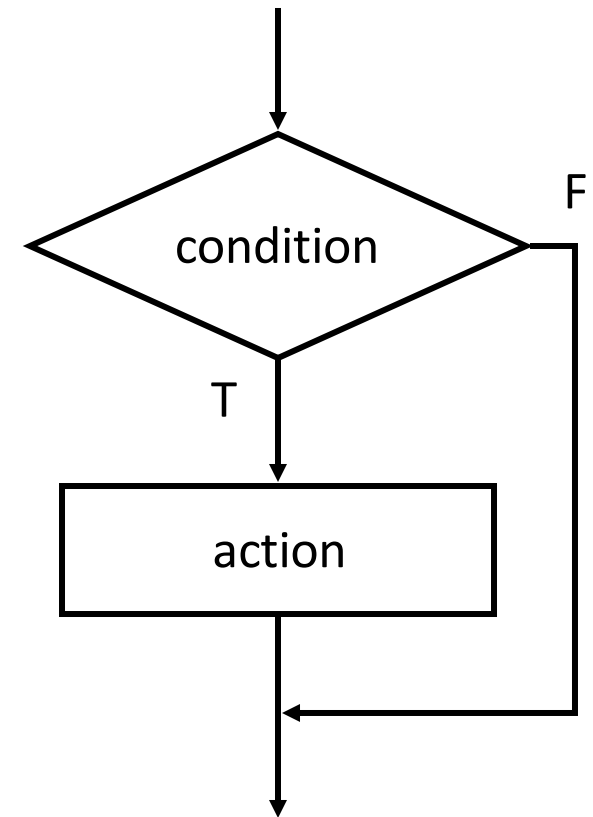- while

- do - while

- for

# The if Statement (similar to BR in LC-3)

```
int x, y;
/* assign some value to x, code omitted */


if (x < 0)
    x = -x;  /* negate x only if x < 0 */



if ((x > 10) && (x < 20)){
    y = x % 4;
    /* What would be the value of y? */
    printf("y = %d\n", y);
}
```
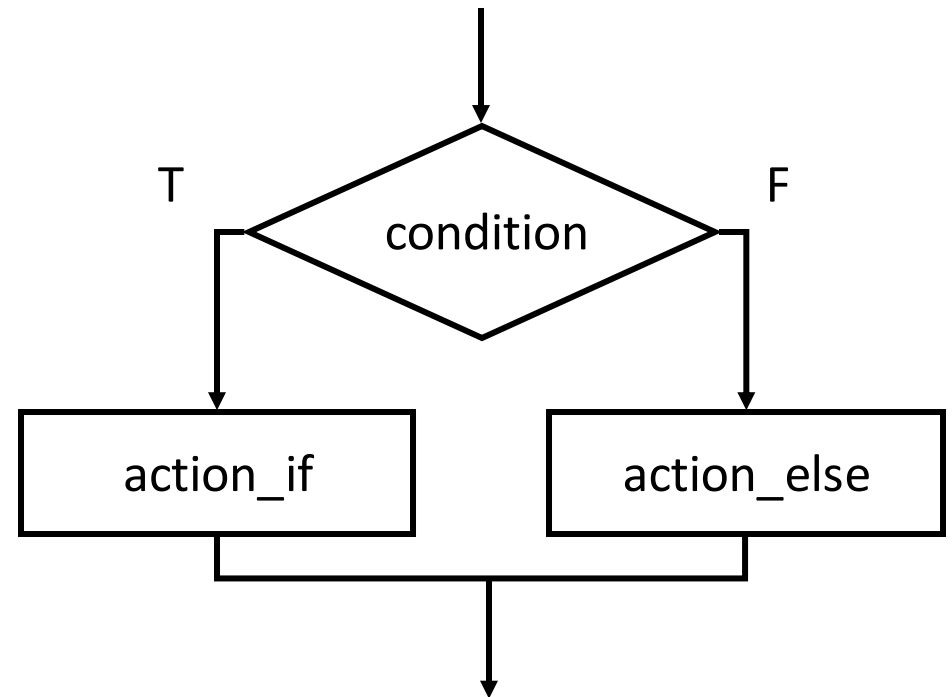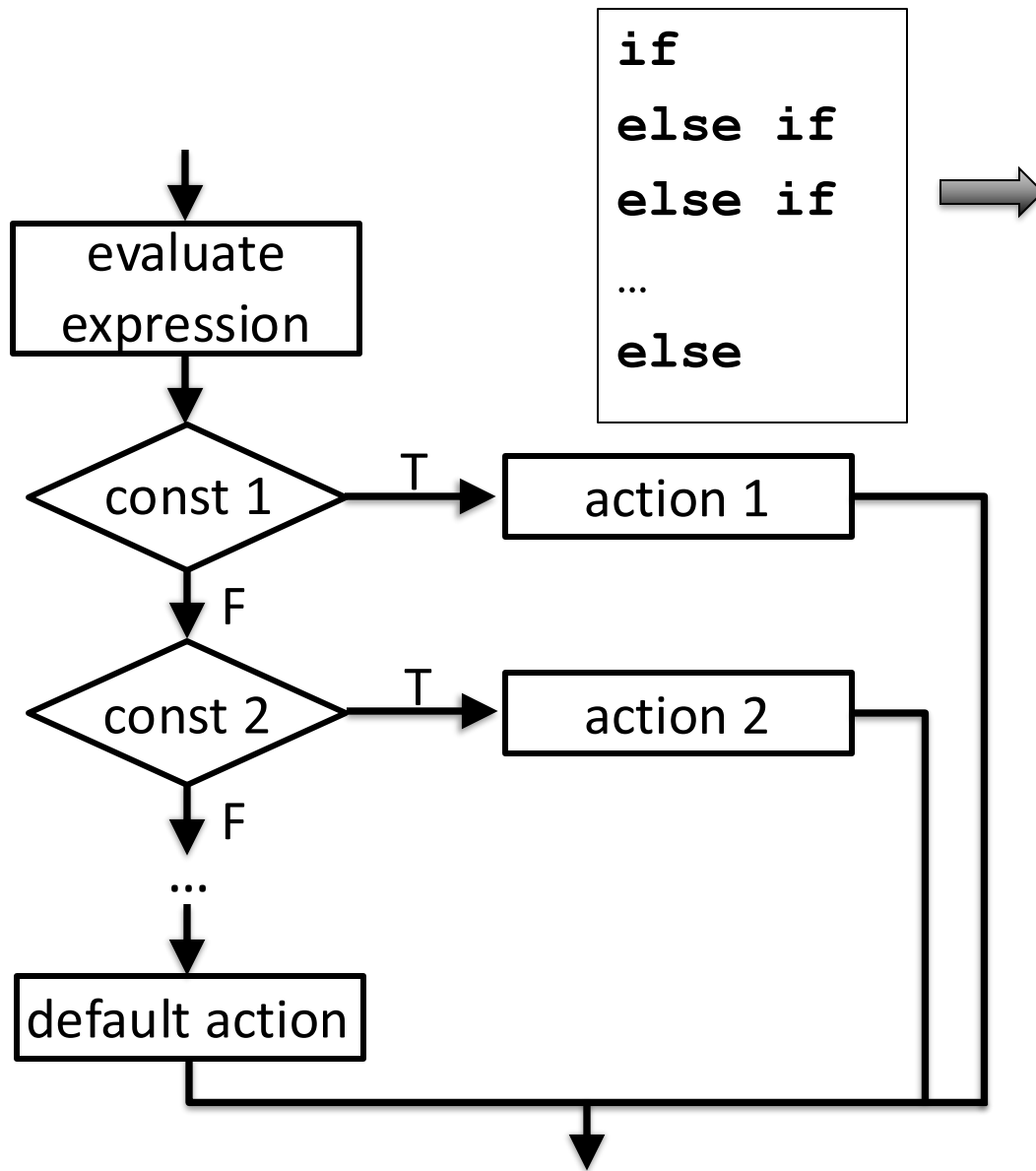
ECE ILLINOIS

# The if - else Statement

```c
int x;
/* assign some value to x,
code omitted */

if (x < 0)
    x = -x;
else
    x = x*x;


if (x%2 == 0)
    printf("x=%d is even\n", x);
else
    printf("x=%d is odd\n", x);
```

ECE ILLINOIS

# The switch Statement



```
if
else if
else if
…
else
```
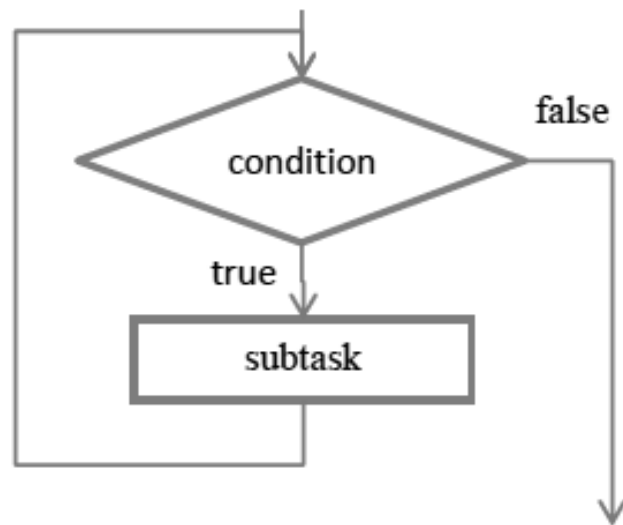
```
switch (expression){
    case const 1:
            action 1;
            break;
    case const 2:
            action 2;
            break;
    ...
    default:
            default action;
            break;
}

/*notice the use of break*/
```
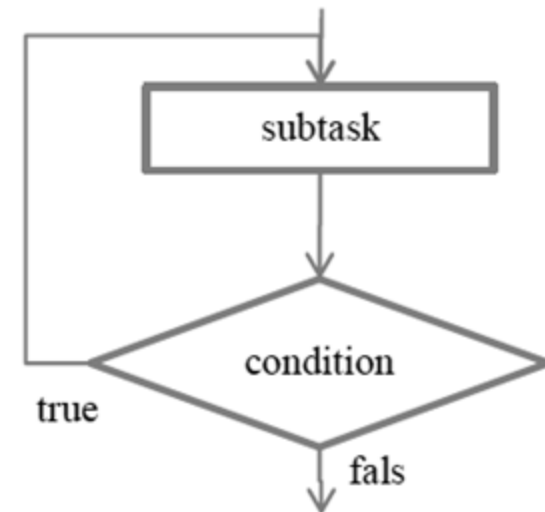
# The while / do - while Statement

while: loop body may or may not
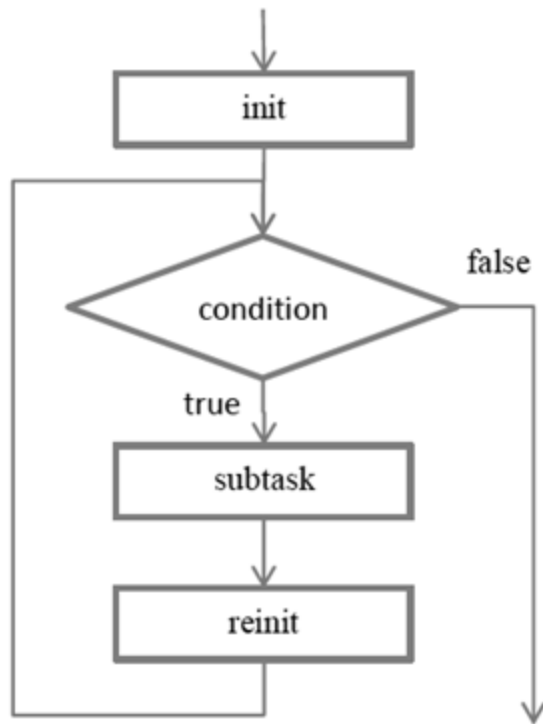be executed even once

do - while: loop body will be
executed **at least once**





```
int x = 0;
while (x < 10) {
    printf("x=%d\n", x);
    x++;
}
```

```
int x = 0;
do {
    printf("x=%d\n", x);
    x++;
} while (x < 10);
```

# The for Statement



```
int x = 0;
while (x < 10) {
    printf("x=%d\n", x);
    x++;
}
```

```
int x;
for (x = 0; x < 10; x++){
    printf("x=%d\n", x);
}
```

➢ What could cause a loop to become **infinite loop**?

```
for (x = 0; x < 10; x++){
    if (x == 5)
        break;
    printf("x=%d\n", x);
}
```

➢ What is the output of the code above? What if **break** is replaced by **continue**?

ECE ILLINOIS

# Nested Loops

inner loop is nested within the outer loop (similar to print hex example in LC-3)

```
for (){
    for (){
        …
    } /* inner loop to shift 4 bits to calculate each digit */
    …
} /* outer loop to print the 4 digits */
```

# Exercise

Write a C program to print an n x n identity matrix using nested loops.

```c
#include <stdio.h>
#define N 5
int main(){



}
```

# Follow-up Questions

➢ What are some ways to <u>stop after printing the third '1'</u> on the main diagonal, such as the example below?

1 0 0 0 0

0 1 0 0 0

0 0 1

➢ How can we <u>take user input</u> for the size of the matrix?

➢ How can we <u>add a check</u> before printing the matrix to ensure user input is within the valid range of $0 < n < 10$? If user input is invalid, print the message "Number entered is invalid" and prompt the user to enter a number again.

ECE ILLINOIS