GNG1106
Fundamentals of Engineering Computation

Instructor: **Hitham Jleed**

University of Ottawa

Fall 2023 ~

In-Class Exercise:

Outline

Outline

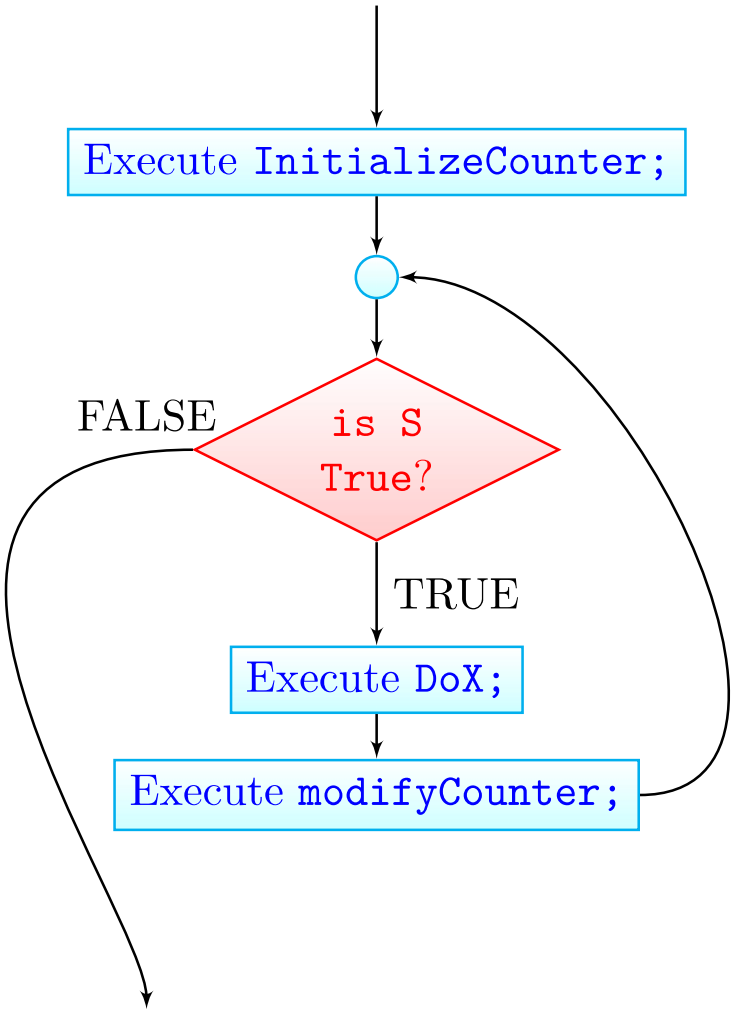
Loops

- It is often required that certain instructions are repeated many times, for example, in the following computation tasks.
  1. Compute the sum $1 + 2 + 3 + \ldots + 100$.
  2. Find the largest number in an array of 100 numbers.
  3. Keep asking the user to enter a value until the entered value becomes invalid.
- The number of repetition rounds may be
  - known to the programmer, as in Examples 1 and 2, or
  - unknown to the programmer, as in Example 3
- There are three ways to implement such "repetitions", knows as loops:
  - for-loop
  - while-loop
  - do-while-loop

for-loop

```
for (initializeCounter; S; modifyCounter)
  DoX;
```

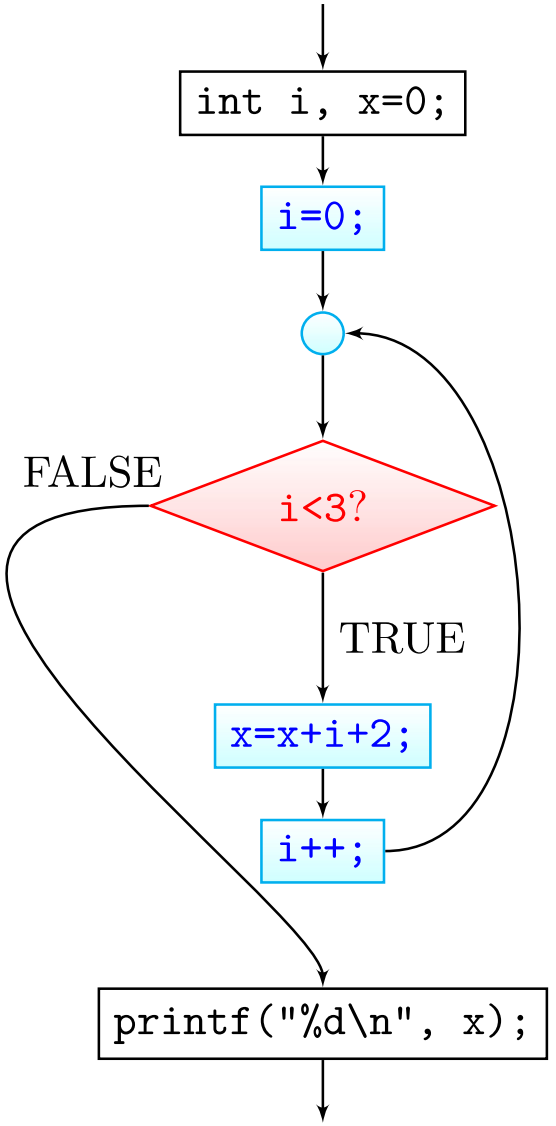
- `initializeCounter` (although syntactically allowed to be any C statement) is typically a statement that assigns an initial value to a (usually integer-typed) variable which serves as the counter.
- `S` is a logical expression that usually (although not necessarily) involves the value of the counter.
- `modifyCounter` (although also allowed to be any C statement) is typically a statement that modifies the value of the counter.
- `DoX` is the code that is to be executed repeatedly. `DoX` is allowed to contain multiple statements, in which case the block of code needs to be enclosed by a pair of curly brackets `{}`.


Execute InitializeCounter;
FALSE
is S
True?
TRUE
Execute DoX;
Execute modifyCounter;

Trace This Code!

```
int i, x=0;
for (i=0; i<3; i++)
   x=x+i+2;
printf("%d\n", x);
```

- `i++` here is the same as `i=i+1`, i.e., increasing `i` by 1.



```
for (initializeCounter; S; modifyCounter)
  DoX;
```

- "`DoX`" is sometimes called the "body" of the loop.
- After "`for (initializeCounter; S; modifyCounter)`", there is no semi-colon "`;`"
  - If you do include a semi-colon there, the compiler won't complain! The compiler simply regards the loop as having an empty body, and takes "`DoX`" as the statement after the for-loop. This of course won't implement the desired repetition.
- The most common/useful form of "`for (initializeCounter; S; modifyCounter)`" is "`for (j=0; j<M; j++)`", which repeats the loop body exactly M times.

Write a Program

Write a program, using a `for-loop`, that computes 1+2+3 + ... + N for an N value entered by the user.

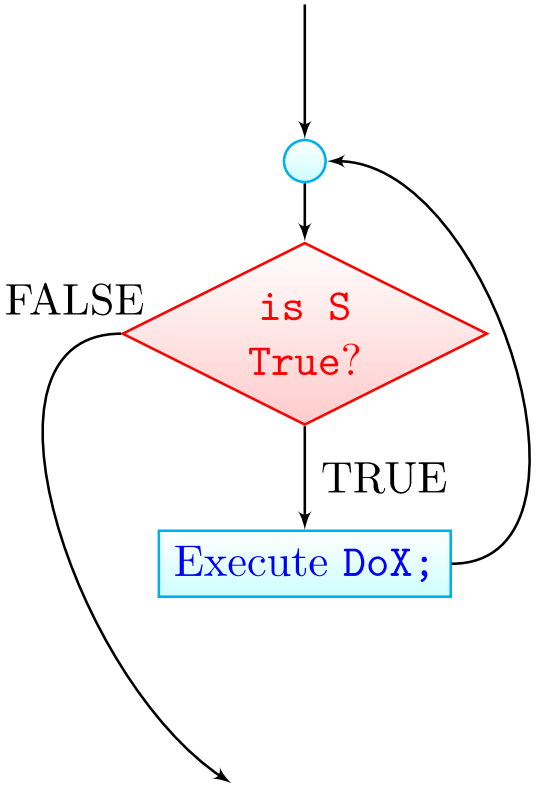
- print the accumulated sum in each iteration

Coding Demonstration

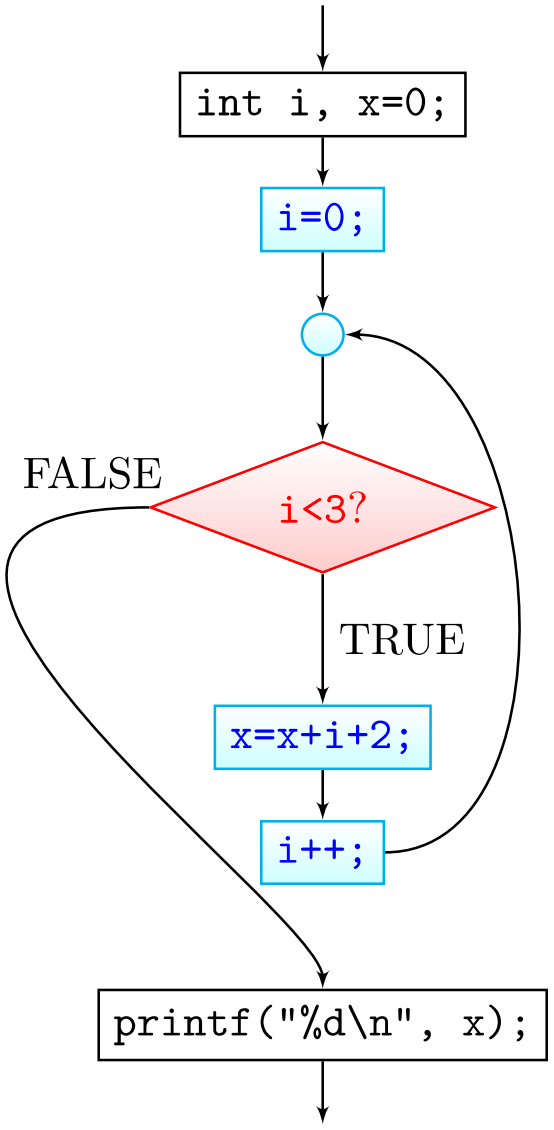
while-loop

```
while (S)
    DoX;
```

- `S` is a logical expression; no semi-colon after `while(S)`.
- `DoX` is the code to be executed repeatedly.
- When `DoX` contains more than one statements, the block of code must be enclosed by a pair of curly brackets `{ }`.
- The program enters the loop if `S` is TRUE, and repeats `DoX` until `S` evaluates to FALSE.
- Before using the while loop, additional code is usually required for the program to enter the loop (i.e., to start repetition).
- Inside `DoX`, something needs to be done for `S` to eventually evaluate to FALSE so that the program can exit from the loop.


FALSE
is S
True?
TRUE
Execute DoX;

- The similarity between for-loop and while-loop is that in both cases, the condition ("is `S` True") is checked before the program enters the loop ("pre-tested").
- Comparing with while-loop, for-loop offers more rigid structure to control the repetition of loop body. Whereas in while-loop, all design logic is left for the programmer.
- Everything that can be implemented with for-loop can be implemented with while-loop.




```
int i, x=0;
for (i=0; i<3; i++)
   x=x+i+2;
printf("%d\n", x);
```

```
int i, x=0;
i=0;
while (i<3)
{
   x=x+i+2;
   i++;
}
printf("%d\n", x);
```

Definite `while`-loop

- A definite while-loop is a while-loop in which the number of repetitions is known to the programmer (e.g., previous example).
- A definite while-loop is typically controlled by a counter, and the following is required for the loop to execute correctly.
  - The counter should be initialized before the loop.
  - There is a statement in the loop body `DoX` that modifies the value of the counter.
  - The logical expression `S` (which tests against the counter) should evaluate to TRUE initially to allow the program to enter the loop and and evaluate to FALSE after the desired number of repetitions.

Indefinite `while`-loop

- An indefinite while-loop is a while-loop in which the number of repetitions is unknown to the programmer.
- An indefinite while loop is typically controlled by a "sentinel", (i.e., a variable not necessary serving as a counter) and the following is required for the loop is to execute correctly.
  - The sentinel should be initialized before the loop.
  - The loop body `DoX` includes a mechanism that modifies the value of the sentinel.
  - The logical expression `S` (which tests against the setinel) should evaluate to TRUE initially to allow the program to enter the loop and and evaluate to FALSE to allow the program exit from the loop.

An Example of Indefinite `while`-loop

```
int sentinel;
sentinel = 1;
while (sentinel != -1)
{
   printf("Enter a value ( -1 to exit the
loop)\n");
   scanf("%d", &sentinel);
}
```

Write a Program

- Write a program, using a `while-loop`, that computes 1+2+3 + ... + N for an N value entered by the user. Print the accumulated sum in each iteration.
- Re-write the "City Hall" program, using a `while-loop`, to allow the program to prompt the user again until a valid option is entered.
- Re-write the "toss a die" program, using a `while-loop`. to allow the user to keep guessing until he gets the die value.

Coding Demonstration