Computer Programming & Problem Solving
CS100

Mrs Sanga G. Chaki
Department of Computer Science and Engineering
National Institute of Technology, Goa
November, 2022

Some new terms

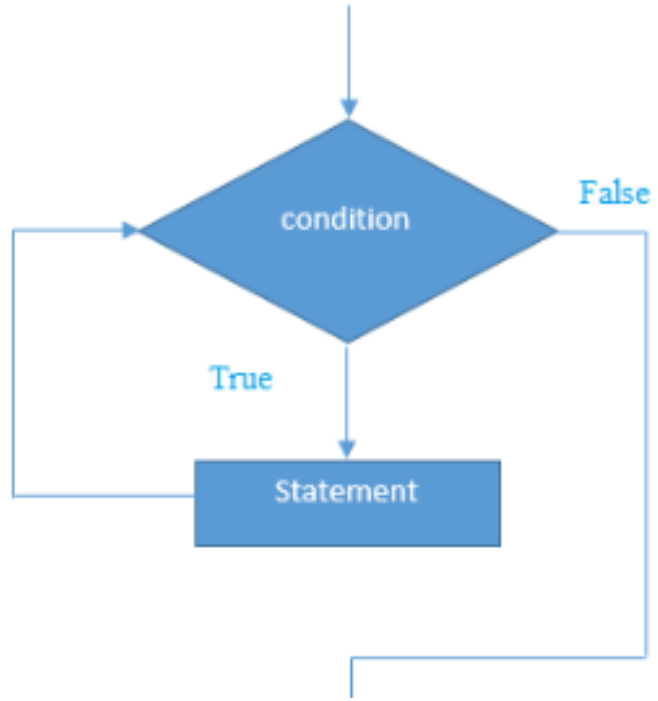
1. Counter
2. Iteration
3. Dry run

Loops

1. Loops: Group of instructions that are executed repeatedly while some condition remains true.
2. While – condition controlled
3. For – counter controlled
4. Do-while – Sentinel controlled
5. There are 4 things that happen in a loop:
   a) Initialize counter – to count number of passes
   b) Check condition
   c) Go through/skip through loop - iteration
   d) Update counter

While Loop

1. while(condition){
   a) statement_1;
   b) ...
   c) statement_N; }
2. The while loop will not be entered if the loop control expression evaluates to false (zero) even before the first iteration.
3. The break statement can be used to come out of the while loop.



While Loop: Sum of first N natural numbers

```
int main () {
    int N, count, sum;
    scanf ("%d", &N) ;
    sum = 0; count = 1;

    while (count <= N)  {
        sum = sum + count;
        count = count + 1;
    }

    printf ("Sum = %d\n", sum) ;
    return 0;
}
```

Increment/Decrement Operator

1. the increment operator ++ increments the value of i by 1, every time the statement i++ gets executed.
2. To reduce the value of a variable by 1 a decrement operator- - is used.
3. i=i+1
4. i++
5. i+=1 are all same
6. Post increment/pre increment

For Loop

1. General form
   a) for(expr1; expr2;expr3)
      - Statement;
2. expr1: initializes loop parameters
3. expr2: test condition, loop continues if this is satisfied
4. expr3: used to alter the value of the parameters after each iteration
5. statement: body of the loop

For Loop: Sum of first N natural numbers

```
int main () {
    int N, count, sum;
    scanf (“%d”, &N) ;

    sum = 0;
    for (count=1; count <= N; count++)
        sum = sum + count;

    printf (“Sum = %d\n”, sum) ;
    return 0;
}
```

Do While Loop

1. The do-while loop looks like this:
    1. do
    2. {
        1. this ;
        2. and this ;
        3. and this ;
        4. and this ;
    3. } while (this condition is true) ;

Do While Loop

1. This means that do-while would execute its statements at least once, even if the condition fails for the first time.
2. main( )
3. {
    1. do
    2. {
    3. printf ("Hello World \n") ;
    4. } while (4 < 1) ;
4. }
5. Do the sum of first N natural numbers using do-while – Home work
6. If while(4>1) → infinite loop

Infinite Loop

1. The loops that we have used so far executed the statements within them a finite number of times
2. An infinite loop is a looping construct that does not terminate the loop and executes the loop forever.
3. It is also called an indefinite loop or an endless loop.
4. It either produces a continuous output or no output.
5. It can be an error → the do while example in previous slide
6. Or by design → when?
7. When to use?
   - when it is not known beforehand how many times the statements in the loop are to be executed
   - All the games. The game will accept the user requests until the user exits from the game.

Infinite Loop – By design

```
/* Execution of a loop an unknown number of times */
main( )
{
      char  another ;
      int  num ;
      do
      {
            printf ( "Enter a number " ) ;
            scanf ( "%d", &num ) ;
            printf ( "square of %d is %d", num, num * num ) ;
            printf ( "\nWant to enter another number y/n " ) ;
            scanf ( " %c", &another ) ;
      } while ( another == 'y' ) ;
}
```

Infinite Loop – By accident

```c
// Online C compiler to run C program online
#include <stdio.h>

int main() {
    int i=0;
    while(i<10){
        printf("Hello\n");
    }
    return 0;
}
```

Break statement

1. We often come across situations where we want to jump out of a loop instantly, without waiting to get back to the conditional test.
2. We have used this in switch case
3. The keyword break allows us to do this.
4. When break is encountered inside any loop, control automatically passes to the first statement after the loop.
5. A break is usually associated with an if.

Break statement - Example

```c
// Online C compiler to run C program online
#include <stdio.h>

int main() {
    int i;
    for (i = 0; i < 10; i++) {
        if (i == 4) {
            break;
        }
        printf("%d\n", i);
    }
    return 0;
}
```

```
/tmp/o4rFTZ4N78.o
0
1
2
3
```

Without Break statement - Example

```c
// Online C compiler to run C program online
#include <stdio.h>

int main() {
    int i;
    for (i = 0; i < 10; i++) {
        if (i == 4) {
            // break;
        }
        printf("%d\n", i);
    }
    return 0;
}
```

```
/tmp/o4rFTZ4N78.o
0
1
2
3
4
5
6
7
8
9
```

Break from accidental Infinite loop

```c
// Online C compiler to run C program online
#include <stdio.h>

int main() {
    int i=0;
    while(i<10){
        printf("Hello\n");
        break;
    }
    return 0;
}
```

```
/tmp/o4rFTZ4N78.o
Hello
```

Continue statement

1. In some programming situations we want to take the control to the beginning of the loop, bypassing the statements inside the loop, which have not yet been executed.
2. The keyword continue allows us to do this.
3. When continue is encountered inside any loop, control automatically passes to the beginning of the loop.

Without Continue statement - Example

```c
#include <stdio.h>
int main() {
    int i, sum=0;
    for(i=1;i<=10;i++){
        // if(i==5){
        //     continue;
        // }
        printf("i = %d\n",i);
        sum = sum+i;
    }
    printf("sum = %d",sum);

    return 0;
}
```

```
/tmp/VmEoOnijz1.o
i = 1
i = 2
i = 3
i = 4
i = 5
i = 6
i = 7
i = 8
i = 9
i = 10
sum = 55
```

With Continue statement - Example

```c
#include <stdio.h>
int main() {
    int i, sum=0;
    for(i=1;i<=10;i++){
        if(i==5){
            continue;
        }
        printf("i = %d\n",i);
        sum = sum+i;
    }
    printf("sum = %d",sum);

    return 0;
}
```

```
/tmp/VmEoOnijz1.o
i = 1
i = 2
i = 3
i = 4
i = 6
i = 7
i = 8
i = 9
i = 10
sum = 50
```

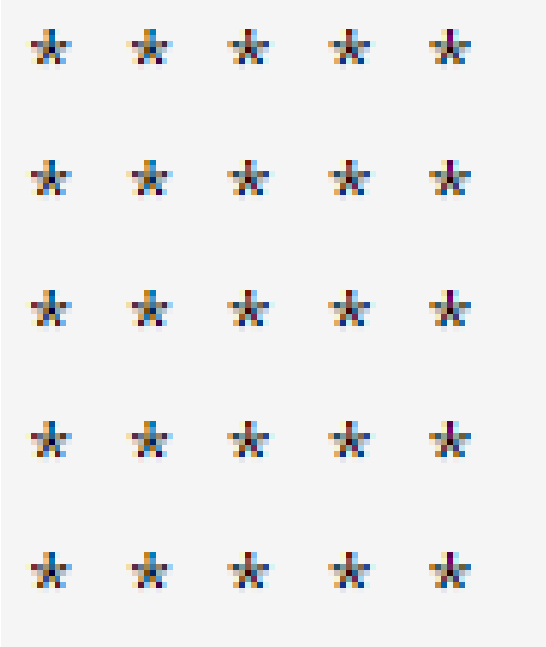
Break and Continue - Example

```c
#include <stdio.h>
int main() {
    int i, sum = 0;
    for(i=1;i<=10;i++){
        if(i%2==0)
            printf("%d\n",i);
        else{
            if(i==9)
                break;
            else
                continue;
        }
    }
    printf("Hello world");
return 0;
}
```

```
/tmp/4hhnDhreoy.o
2
4
6
8
Hello world
```

Nesting of Loops

1. We can nest loops
2. Printing 2D figures on screen




```c
int main() {
    int i,j;
    for(i=0;i<5;i++){
        for(j=0;j<5;j++){
            printf("* ");
        }
        printf("\n");
    }

    return 0;
}
```

Nesting of Loops

```
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * * * *
* * * * * * * * *
```

```c
int main() {
    int i,j;
    for(i=0;i<10;i++){
        for(j=0;j<i;j++){
            printf("* ");
        }
        printf("\n");
    }

    return 0;
}
```