UESTC 1005 - Introductory Programming

Lecture 5 - Loops 🎠

Hasan T Abbas

Hasan.Abbas@glasgow.ac.uk

Lecture Outline

- The need for iteration
- The `while` loop
- The `for` loop
- Nested Loops

Iteration means repetition of a process

Loop is a process which is connected to its beginning

Why iterations ❓

In the last lecture, we talked about *building* logic.
Algorithms build on logic to make *logical and meaningful* decisions.

An **algorithm** is a sequence of clearly defined steps and instructions with clear start and end points.

Iterations let us automate boring tasks

Loops in C

- Statements that are executed *repeatedly*

- Every loop has a *controlling expression*

- C has three kinds of loops
    i. `while`
    ii. `do while`
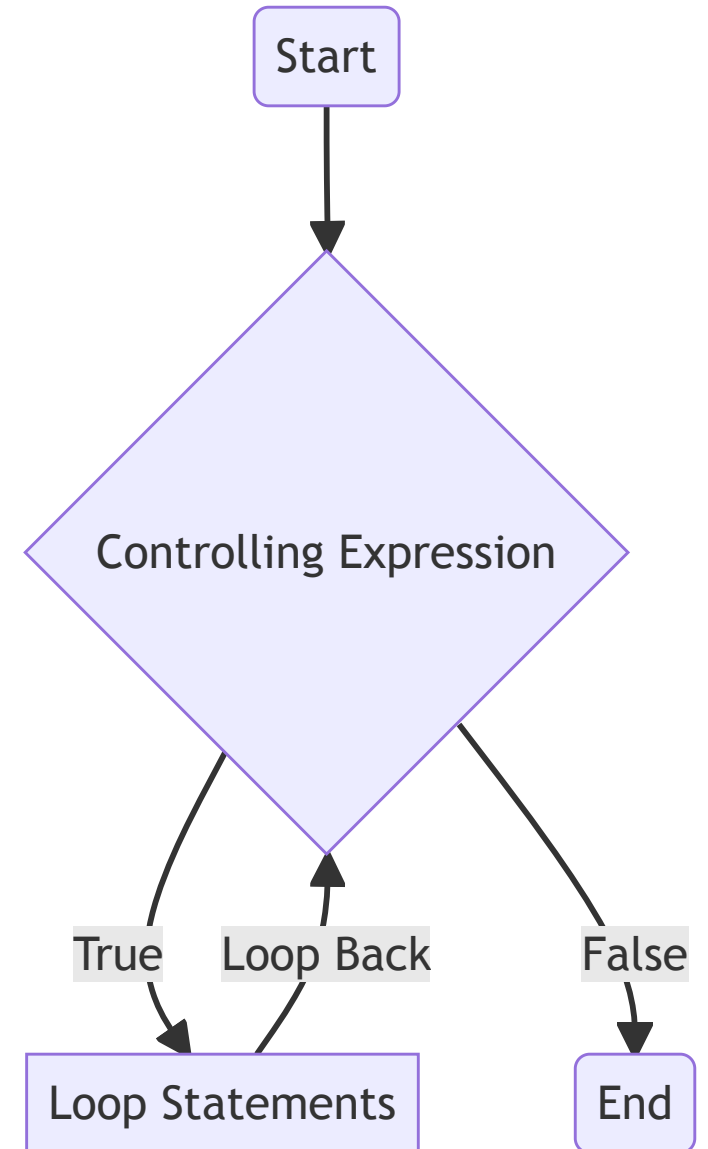   iii. `for`

The `while` loop

`while` is the simplest of the loops

- The controlling expression is executed
  before `the loop block.

```
while (controlling expression)
{
    // statements
}
```

- Note there is no `;` in the first line
- Just like before `controlling expression`
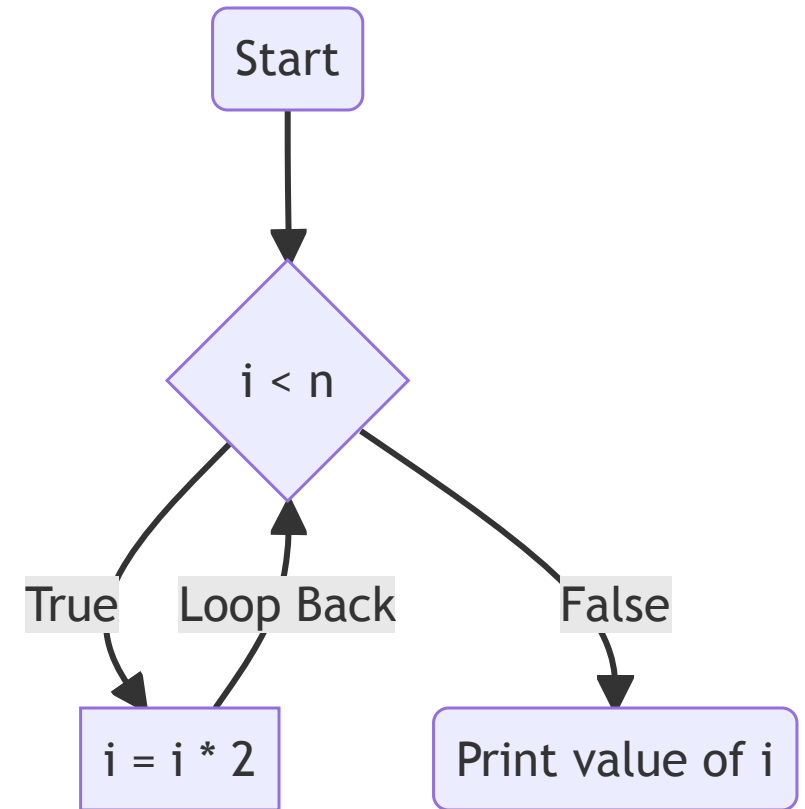 can be `true` or `false`

Start

Controlling Expression

True   Loop Back   False

Loop Statements   End

Example - `while` loop 🪧

Write a C program that calculates a number equal to the smallest power of 2 that is greater than or equal to 20 .

$$x = 2^i \geq n$$

```c
int i, n;
i = 1; n = 20;
while (i < n)
{
    i = i * 2;
}
printf("The smallest value is %d \n", i);
```

Start

i < n

True   Loop Back   False

i = i * 2   Print value of i

How Example worked 🪧

```
i = 1; i is now 1.
Is i < n? Yes; continue.
i = 1*2; i is now 2.
Is i < n? Yes; continue.
i = i*2; i is now 4.
Is i < n? Yes; continue.
i = i*2; i is now 8.
Is i < n? Yes; continue.
i = i * 2; i is now 16.
Is i < n? Yes; continue.
i = i * 2; i is now 32.
Is i < n? No; exit from loop.
```

The *infinite* Loop

Sometimes we delibrately construct a condition that remains *true* all the time.

- When paired with a loop, we call this an infinite loop

- It executes forever until we tell the compiler to stop

- We can use `break` or some other ways to get out of the loop

- We commonly use infinite loops in embedded systems such as an Arduino.

```
while(1)
{
    // some statements inside
}
```

Example - Infinite Loop 🪧

Write a C program that accepts integers from a user through the keyboard and then computes their sum. The process is carried on *until* the user enters `0`

Program Template:

```
A C program that computes the sum of non-zero integers

Enter an integer (0 to terminate):
12
32
345
-1
-10
0

The sum of the entered integers is: 378
```

# Example - Infinite Loop Implementation 🪧

```c
#include<stdio.h>
int main(){
    // we are going to define an infinite loop
    int sum, input;
    sum = 0; // always initialise to 0 as the initial value may be undefined
    printf("Enter an integer (0 to terminate):\n");
    while(1){
        scanf("%d", &input);
        if (input == 0){
            break;
        }
        else{
            sum += input;
        }
    }
    printf("The sum of the entered integers is: %d\n", sum);
    return 0;
}
```

The `do while` Loop

- We check the condition after the body statements are executed

```
do{
    // Statements
} while ( expression);
```

- Note the `;` in the end of the `while()`

- `do while` loops run at least once

```
Start
  |
  v
i = i * 2
  |
  v   True
i < n
  |
  | False
  v
Print value of i
```
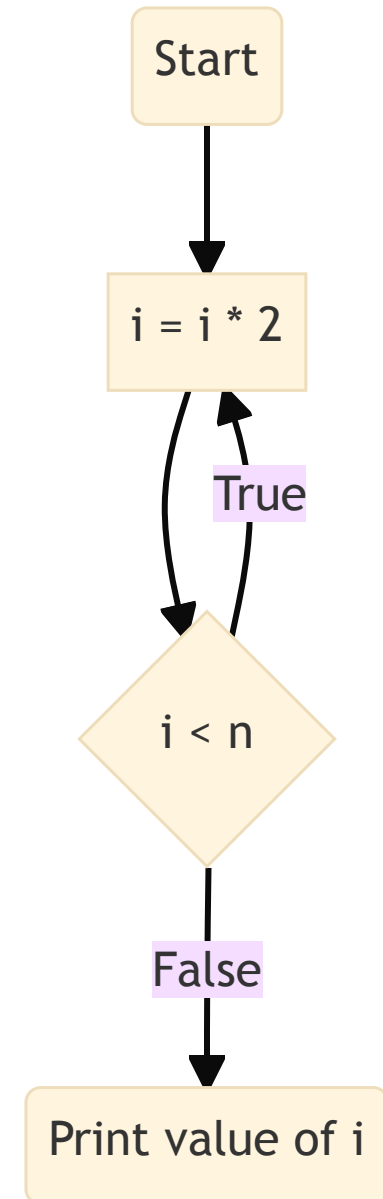
The `for` Loop

- `for` loop is the most C-like thing in C

- Best way to write loops

- We have a *counting variable* that dictates the iterations

- Ideal for cases where we need to count *up* or *down*

```
for( expr1 ; expr2 ; expr3){
    //statements
}
```

for (expr1 ; expr2 ; expr3)

Initialise Counting variable

Check condition

Change counting variable value

14
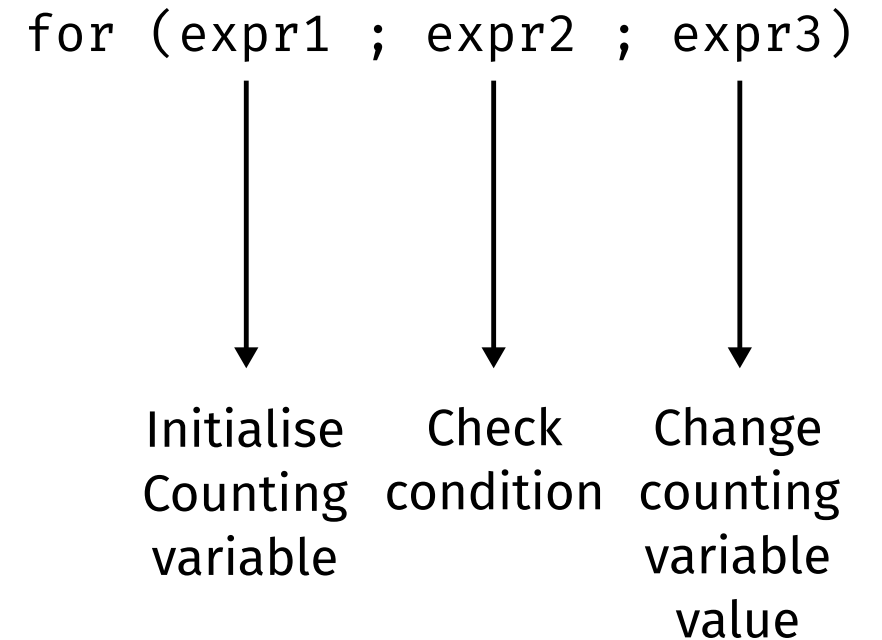
`for` Loop Examples

Mixing Operators

- Create a `for` loop to *even* display numbers from 2 to 200.

```
for (i = 2 ; i <= 200 ; i += 2){
    printf("%d\n",i);
}
```

Countdown!

- Create a counter down from `n - 1` to `0`.

```
for (i = n - 1 ; i >= 0 ; i--)
```

- Create a counter down from `n` to `1`.

```
for (i = n  ; i > 0 ; i--)
```

Flexible `for` loop

- The three expressions in the `for` loop declaration are *optional*
- Something like:

```
for (i = 0 ; i < 2 ;)
```

or

```
for(; i < 20 ; )
```

or

```
for (; ;)
```

are all valid in C

Quiz ❓

Pick a `for` loop statement that counts from `1` to `100` in steps of `10` .

Go to

https://www.menti.com/al4htqpzusws

and type the code `3874 5754` .

⚠️ Some Care needed

Modern versions of C allow us to *declare* a variable in the first `for` *expression*

```c
for (int i = 0; i < 20 ; i++){
    printf("%d",i); // -----> RIGHT
}
printf("%d",i); // ----> WRONG
```

The variable `i` is only visible *inside* the loop. It cannot be accessed from outside.

20

Example - Display a Table of Cubes 🪧

Say we want to compute the cube of the first 10 integers and display on the screen. Lets use `for` loop.

```c
for (int n = 0, cube = 0; n <= 10 ; n++){
    cube = n * n * n;
    printf("The cube of the number %8d is %8d \n", n, cube);
}
```

Points to note

- The comma `,` operator lets us glue two or more expressions
- `%8d` is called a *placeholder* where we reserve 8 characters on the screen to display an integer value

Control Statements - `break`

- `while` and `for` loops have exit points typically before the body

- Sometimes we require to exit in the middle of the code

- Just like with `switch` the `break` statement takes us out of the loop

```c
for (i = 2; i < n; i++){
    if (n % i == 0){
        break;
    }
    else if (n < i){
        printf("%d is divisible by %d \n", n, i);
    }
    else{
        printf("%d is prime \n", n);
    }
}
```

Control Statements - `continue`

- Unlike `break`, the `continue` statement doesnt really take us out of the loop.

- Rather, it skips the current iteration of the loop.

```
for(int n = 0, sum = 0; n < 10; n++){
    if (n == 3 || n == 6 || n == 9){
        continue;
    }
    sum += n;
}
```
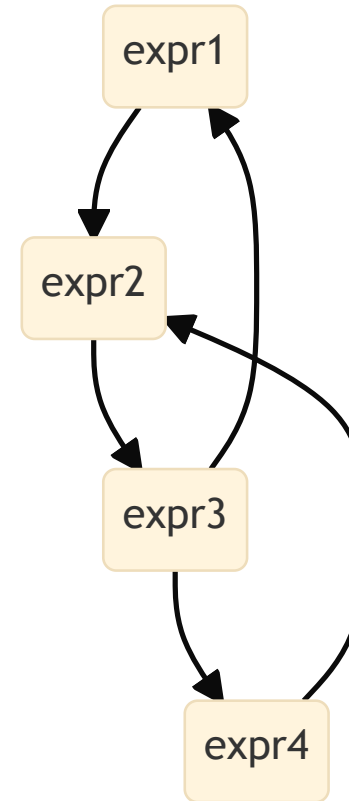
The above code computes the series:

$$1 + 2 + 4 + 5 + 7 + 8$$

Control Statements - `goto`

- `goto` is an *unconditional* jump statement in the program

- All we need is to create a `label`

- `Labels` are denoted by a `:` and written before a statement

```c
for (i = 2; i < n; i++)
        if (n % i == 0)
            goto done;
done:   if (i < n){
        printf("%d is divisible by d\n", n, d);
    }
else{

        printf("%d is prime\n", n);

    }
```

- Results in a haphazard, *spaghetti* code

Quiz ?

What is the output of the program below:

```c
#include <stdio.h>
int main()
{
    int sum = 0;
    for (int n = 0; n < 10; n++){
        if (n == 0){
            break;
        }
        sum += n;
    }
    printf("The sum is: %d\n", sum);
    return 0;
}
```

Visit

https://www.menti.com/al4htqpzusws

and type the code 3874 5754 .

Today's Summary 🗒️

- Introduced three types of loop statements

- Looked into control statements

Next up 📅

- Nested Loops 👓
- Functions
- Recursion 🪧

Questions ❓

https://www.menti.com/al4htqpzusws
and type the code `3874 5754` .