If Statement & While Loop

If Statement

An if statement is a function used to allow a program to choose which code to execute and which to ignore. It does this by checking if the given argument results in a value of 1 (TRUE) or 0 (FALSE) and executing different code accordingly. Similar to `printf()` or `scanf()`, the if statement does not need to be declared or have its return value written because it is built into the C language.

Here are the syntaxes of the if statement in C:

`if (argument) { code to be executed }`

This is the most commonly used syntax. If the argument results in a value of 1, the code within the curly braces will be executed; otherwise, if it's 0, the code will not be executed.

`else if (argument) { code to be executed }`

This syntax works the same way as `if()`, but it is only used if you have already created an if statement and want to add a different option for the program. `else if()` can be created multiple times to give the program more diverse options.

`else { code to be executed }`

This syntax is only used if you want to provide a final option for the program. Note that `else` does not require an argument, so if none of the previous if statements are met, only the code within `else` will be executed.

Below is an example program that compares two numbers:

```
#include <stdio.h>


int main(void) {


    int angka1 = 1;
    int angka2 = 2;


    if (angkal < angka2) {
        printf("angkal lebih kecil dari angka2. \n");
    } else if (angkal == angka2) {
        printf("angkal sama dengan angka2.\n");
    } else {
        printf("angkal lebih besar dari angka2.\n");
    }


}
```

Output:

```
C:\Users\Desktop\new 1.exe


angkal lebih kecil dari angka2.


Process exited after 0.05168 seconds with return value 0
Press any key to continue
```

While Loop

A while loop is a function used to execute the same code repeatedly. The while loop will continue to repeat the code execution as long as the given argument results in a value of 1 (TRUE) or more. When the argument results in a value of 0 (FALSE), the program will stop looping and execute the next code. Like the if statement, the while loop is also built into the C language, so it does not need to be declared or have its return value written.

```
while(condition) {
    // Code to be executed repeatedly
}
```

For example, the code within the while loop in this program will be executed repeatedly as long as `n <= 10`. In each loop, the value of `n` is incremented by 1, so there will be a condition where `n > 10`.

```
#include <stdio.h>

```

```
int main(void) {

    int n = 1;

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

    return 0;
}
```

Output:

```
n = 1
n = 2
n = 3
n = 4
n = 5
n = 6
n = 7
n = 8
n = 9
n = 10

Process exited after 0.04058 seconds with return value 0
Press any key to continue
```

It is also important to know that a while loop can repeat execution indefinitely if the argument never changes, in other words, it always has a value of 1 or more.

```
#include <stdio.h>

int main(void) {

    while (1) {
        printf("infinite loop! ");
    }

    return 0;
}
```

Output:

```
infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite
loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop!
infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite
loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop!
infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite
loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop!
infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite
loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop! infinite loop!
infinite loop! infinite loop!...
```

Revision #1
Created 19 February 2025 13:13:06 by GI
Updated 19 February 2025 13:27:15 by GI