C Programming

Conditional statements

In C and in many other languages, choices can be made in terms of code execution

The if statement

The if statement is as follows:

```
if (condition)
{
        statement 1;
        …
        …
        statement n;
}
```

The condition **must** be true in order for the code statements inside the curly brackets to execute. If the condition is false, the code statements inside the curly brackets are ignored.

e.g.

```
        float bank_balance = 0;

        // check if bank balance is greater than zero
        if (bank_balance > 0)
        {
                printf("You have money");
}
```

Note: Always use { and } braces around the statements inside the if statement. Even though the if statement allows you to omit the { and } when there is only 1 statement to execute, I highly advise you use them in case you need to return and insert additional statements inside the { and }. This avoids any potential bugs.

```
/*
Program to demonstrate the if statement
*/
#include <stdio.h>

int main()
{
```

```
	float bank_balance = 0;

	printf("Enter a bank balance\n");
	scanf("%f", &bank_balance);

	// if the bank balance is greater than zero
	if (bank_balance > 0)
	{
		printf("\nYou have money");
	}

	// if the bank balance is less than or equal to zero
	if (bank_balance <= 0)
	{
		printf("\nYou have no money - go study Computer Science");
	}

	return 0;
}
```

Repl 4.1: https://replit.com/@michaelTUDublin/41-Simple-if#main.c

Operator	Meaning
==	Equivalent to / Equal to
!=	Not equivalent to / Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

The if .. else statement

The if..else statement is as follows:

```
if (condition)
{
        statement 1;
        …
        …
        statement n;
}
else
{
        statement 1;
        …
        …
        statement n;

}
```

The if .. else statement gives you a choice of executing one or the other lines of code

```
/*
Program to demonstrate the if..else statement
*/
#include <stdio.h>
int main()
{
      float bank_balance = 0;

      printf("Enter a bank balance\n");
      scanf("%f", &bank_balance);

      // if the bank balance is greater than zero
      if (bank_balance > 0)
      {
            printf("\nYou have money");
      }
      else
      {
            printf("\nYou have no money - go study Computer Science");
```

```
    }

    return 0;
}
```

Repl 4.2: https://replit.com/@michaelTUDublin/42-if-else#main.c

The if .. else statement BUT with an added extension

You can extend the if .. else statement by having extra if .. else's associated with the overall code

e.g.,

```
if (condition_1)
{
        statement 1;
        …
        …
        statement n;
}
else if (condition_2)
{
        statement 1;
        …
        …
        statement n;
}
else
{
        statement 1;
        …
        …
        statement n;
}
```

For example:

```
/*
Program to demonstrate the if..else statement
*/
#include <stdio.h>

int main()
```

```
{
	float bank_balance = 0;
	printf("Enter a bank balance\n");
	scanf("%f", &bank_balance);

	// if the bank balance is greater than zero
	if (bank_balance > 0)
	{
		printf("\nYou have money");
	} // end if
	else if (bank_balance == 0)
	{
		printf("\nYou have a zero balance");
	}
	else
	{
		printf("\nYou are in negative balance");
	} // end else

	return 0;

} // end main
```

Repl 4.3: https://replit.com/@michaelTUDublin/43-if-else-if-else#main.c

Logical Operators

The C programming language provides logical operators to use with if / if .. else statements

Logical Operator	Meaning
&&	AND
||	OR
!	NOT

```c
/*
Program to demonstrate logical operators
*/
#include <stdio.h>

int main()
{
	float bank_balance = 0;

	printf("Enter a bank balance\n");
	scanf("%f", &bank_balance);

// if the bank balance is greater than zero AND less than or equal to zero
	if (bank_balance > 0 && bank_balance <= 100)
	{
		printf("\nYour balance is between 1 and 100");
	} // end if

	return 0;

} // end main()
```

Repl 4.4: https://replit.com/@michaelTUDublin/44-Logical-AND#main.c

Using the NOT logical operator, this is as follows:

```c
	int a = 1;

	if ( ! (a == 0) )   // same as: if (a != 0)
	{
		printf("a is NOT zero");
	}
```

Nested if statements

Nested if statements are if statements within another if statement

example:

```
/*
Program to demonstrate nested if statements, i.e., if inside if
*/
#include <stdio.h>

int main()
{
	float bank_balance = 0;
	printf("Enter a bank balance\n");
	scanf("%f", &bank_balance);

	/*
	// if the bank balance is greater than zero
	if (bank_balance > 0 && bank_balance <= 100)
	{
		printf("\nYour balance is between 1 and 100");
	} // end if
	*/

	// This code is the same as the if statement above
	if (bank_balance > 0)
	{
		if (bank_balance <= 100)
		{
			printf("\nYour balance is between 1 and 100");
		} // end inner if

	} // end outer if

 return 0;

} // end main()
```

Repl 4.5: https://replit.com/@michaelTUDublin/45-Nested-if#main.c

Advice: Do not nest if statements too deep, i.e., an if statement within an if statement within an if statement, etc., This is because it is very easy to create a bug that is difficult to locate and resolve.

I advise you keep the nesting to a max of 3 levels. If you need to nest any deeper than 3 levels, it is better to redesign the program.

The switch statement

The **switch** statement is used as an alternative to multiple if .. else statements.

Let's say we have the following:

```
int main()
{
	char grade = '  '.

	printf("Enter a grade\n");
	scanf("%c", &grade);

	if (grade == 'A')
	{
		printf("\n Highest grade");
	}
	else if (grade == 'B')
	{
		printf("\n 2nd Highest grade");
	}
	else if (grade == 'C')
	{
		printf("\n 3rd Highest grade");
	}
	…
	…
```

The above code is not very efficient given that it has many if .. else blocks. This uses up CPU cycles and is expensive in processing time. The switch statement can be used as an alternative, which is much more efficient.

Let's take a look at the following example using a switch statement:

```
/*
Program to demonstrate the use of the switch statement
*/
#include <stdio.h>
int main()
{
	char oper;
	float num1, num2, answer;

	printf("Enter a simple maths operation\n");
	scanf("%f", &num1);
```

```
	scanf("%c", &oper);
	scanf("%f", &num2);

	// Now use the switch statement
	switch(oper)
	{
		case '+':
		{
			answer = num1 + num2;
			printf("\n%.1f PLUS %.1f is %.1f", num1, num2,
answer);
			break;

		} // end case '+'

		case '-':
		{
			answer = num1 - num2;
			printf("\n%.1f MINUS %.1f is %.1f", num1, num2,
answer);
			break;

		} // end case '-'

		case '*':
		case 'x':
		case 'X':
	// similar to: if (oper == '*' || oper == 'x' || oper == 'X')
		{
			answer = num1 * num2;
			printf("\n%.1f MULTIPLY %.1f is %.1f", num1, num2,
answer);
			break;

		} // end case '*'

		case '/':
		{
			answer = num1 / num2;
```

```
                printf("\n%.1f DIVIDE %.1f is %.1f", num1, num2,
answer);
                break;

            } // end case '/'

            default:
            {
                printf("\nInvalid operator entered");
                break;

            } // end default

      } // end switch

      return 0;

} // end main
```

Repl 4.6: https://replit.com/@michaelTUDublin/46-Switch-statement#main.c

Programming pitfalls

You should be aware and avoid the following

1. There is no semi-colon at the end of an if statement

e.g.

```
if (bank_balance == 0);
{
        printf("You have a zero balance");
}
```

2. Always place a { and } brace around the if, if ..else, case block code when there is even just 1 line of code / 1 statement of code

```
int a = 0;
int b = 0;
```

```
if (a == 0 && b == 0)
{
        printf("Both a and b are zero\n");
        printf...
        printf...

} // end if
```

3. Be careful you do not try to do too much inside an if / if .. else statement

e.g.

```
int a = 0;
int b = 2;

if (a && b == 0) // This is wrong if (a == 0 && b == 0) This is correct
{
        printf(" ……....");
} // end if
```