

# Exercise 3-1: `if` statements

Maximilian Fernaldy - C2TB1702

## `abs.c`

### Exercise 3-1: `if` statements

15

#### ■ `abs.c`:

- Create a program that displays the absolute value of the entered real number (Tips: `if` statement)

#### ■ `divisor.c`

- Create a program that determines whether B is a divisor of A for the two input integers A and B.
  - When B is a divisor of A, "B is a divisor of A." is displayed.
  - When B is not a divisor of A, "B is not a divisor of A." is displayed.

(Tips: `if - else` statement)

#### ■ `rank.c`

- Create a program that distinguishes ratings of A(excellent) / B(great) / C(good) / D(bad) from the entered points and displays them. Judgment should be made as follows.
  - The score is an integer from 0 to 100
  - 0 ~ 59 → D / 60 ~ 74 → C / 75 ~ 84 → B / 85 ~ 100 → A

(Tips: `if - if else - else` statement)

The absolute value of a real number can be defined as such:

$$|n| = \begin{cases} n, & \text{for positive } n \\ -n, & \text{for negative } n \end{cases}$$

$|n| = \{n, \text{ for positive } n; -n, \text{ for negative } n\}$

which means we can use the conditional operator `if` and `else` to get the final result.

```
#include <stdio.h>

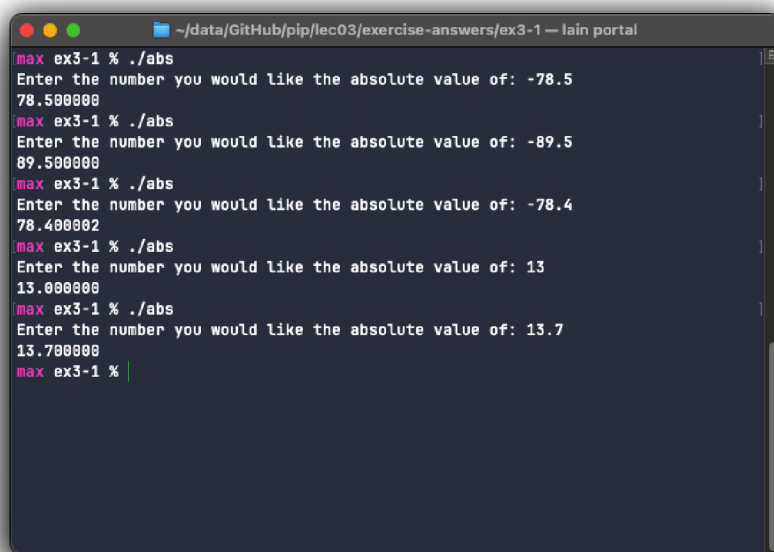
int main() {
    float n = 0;
    printf("Enter the number you would like the absolute value of: ");
    scanf("%f", &n);
    if (n < 0) { // if n is negative, reverse the sign
        printf("%f\n", -n);
    } else { // if n is positive, leave it be
        printf("%f\n", n);
    }
}
```

another, perhaps simpler way to do this is to just use a single `if` operator, reversing the sign if it's negative:

```
#include <stdio.h>

int main() {
    float n = 0;
    printf("Enter the number you would like the absolute value of: ");
    scanf("%f", &n);
    if (n < 0) { // if n is negative, reverse the sign
        n = -n;
    }
    printf("%f", n);
}
```

The program runs as expected:



```
max ex3-1 % ./abs
Enter the number you would like the absolute value of: -78.5
78.500000
max ex3-1 % ./abs
Enter the number you would like the absolute value of: -89.5
89.500000
max ex3-1 % ./abs
Enter the number you would like the absolute value of: -78.4
78.400002
max ex3-1 % ./abs
Enter the number you would like the absolute value of: 13
13.000000
max ex3-1 % ./abs
Enter the number you would like the absolute value of: 13.7
13.700000
max ex3-1 %
```

## divisor.c

In divisor.c we are faced with a binary problem—either B is a divisor of A or it is not. In situations like this we can apply the `if` and `else` conditional operators to return the correct result.

```
#include <stdio.h>

int main() {
    int a = 0, b = 0;
    printf("Input 2 integers A and B separated by a space: ");
    scanf("%d %d", &a, &b);
    if (a % b == 0) {
        printf("B is a divisor of A\n");
    } else {
        printf("B is not a divisor of A\n");
    }
}
```

To check whether or not B is a divisor of A we can use the modulo operator. It returns the remainder of a division. If it returns zero, then there is no remainder, which means B is a divisor of A, and if it returns anything else, then B is not a divisor of A.

## rank.c

For rank.c, we have multiple possibilities to account for. There are 4 possible ratings that can be assigned to a score. To do this we can make use of `if`, `else if` and `else` operators. In this use case, the usage of `else if` is not mandatory, but it is good practice to use it when the possibilities are disjoint (in this case, we cannot get 2 different ratings simultaneously).

```
#include <stdio.h>

int main() {
    int score = 0;
    printf("Enter student score: ");

    scanf("%d", &score);

    if (score >= 0 && score <= 59) {
        printf("D");
    } else if (score >= 60 && score <= 74) {
        printf("C");
    } else if (score >= 75 && score <= 84) {
        printf("B");
    } else if (score >= 85 && score <= 100) {
        printf("A");
    } else {
        printf("Not in range. Please enter integers from 0 to 100.");
    }

    return 0;
}
```

The program is fairly straightforward. It takes the score input by the user and categorizes it into 5 different cases. First, if the score is between 0 and 59 inclusive, the program prints a D rating and ends. If it's between 60 and 74 inclusive it prints a C rating and ends. This also applies similarly to B and A, exiting after assigning a rating, because once the program knows where the score is on the scale, it doesn't need to check again if it belongs to another score range. For example, if the score input is 78, the program will first check if it is between 0 and 59, and return false. Then it will check if it is between 60 and 74 and return false again. It will finally check if it is between 75 and 84, and return true. The program doesn't need to check if it's between 85 and 100 anymore, because it's impossible to be true—we already know that it's between 75 and 84. This is why we should use `else if` instead of another `if` statement. Even though we know for sure that the last `if` statement won't return true, it is a waste of time and resources to check needlessly. In a program with a large amount of conditional operators, we should minimize the amount of checks as much as possible, and only use the correct ones where necessary.

The compiled program works as follows:

```
~/data/GitHub/pip/lec03/exercise-answers/ex3-1 —...
max ex3-1 % ./rank
Enter student score: 59
[D
max ex3-1 % ./rank
Enter student score: 60
[C
max ex3-1 % ./rank
Enter student score: 74
[C
max ex3-1 % ./rank
Enter student score: 75
[B
max ex3-1 % ./rank
Enter student score: 84
[B
max ex3-1 % ./rank
Enter student score: 85
[A
max ex3-1 % ./rank
Enter student score: 100
A
max ex3-1 % ./rank
Enter student score: 101
Not in range. Please enter integers from 0 to 100.
max ex3-1 % ./rank
Enter student score: 0
D
max ex3-1 % ./rank
Enter student score: -1
Not in range. Please enter integers from 0 to 100.
max ex3-1 %
```