

Exercise 3-2: switch statement

Maximilian Fernaldy - C2TB1702

Exercise 3-2: switch statement

16

■ operator.c

- Create a calculator program that performs the four arithmetic operations (+, -, *, /) of the two input real numbers, referring to sample.c in the first lecture.

- Input and output examples:

- Input "12 + 3" → Display: "15.00"
- Input "7.5 - 10" → Display: "-2.50"
- Input "2 * 5" → Display: "10.00"
- Input "10 / 2.5" → Display: "4.00"

- Tips

- The operators of the four arithmetic operations are read as a char type character, and the processing is switched depending on the value in the switch statement.
- Read 3 inputs from the console

```
scanf("%? %? %?", &x, &op, &y);
```

↑ ↑ ↑
What should we choose for the format specification?

The variable op is of type char
'+', '-', '*', '/' are included

Exercise 3-2 introduces another way to deal with multiple possibilities: the `switch` statement. A switch statement is **much faster** than an if-else ladder and is more readable when there are many possibilities to the value of something.

To implement this in operator.c, we first need to identify which variable has multiple possible values. Since we are tasked to create a simple arithmetic calculator, it is obvious that the operator here is that variable. We need to create a **switch** for the operator, which has multiple **cases**: plus sign for addition, minus sign for subtraction, asterisk for multiplication and slash for division.

```
#include <stdio.h>

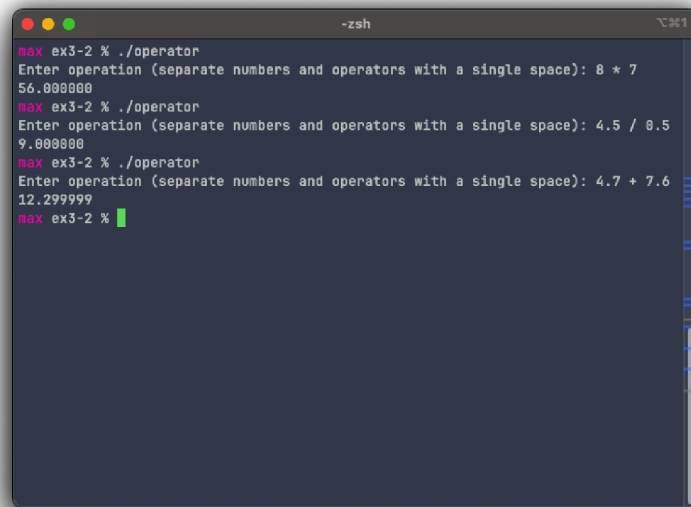
int main() {
    float a = 0, b = 0;
    char opr;
    printf("Enter operation (separate numbers and operators with a single space): ");
    scanf("%f %c %f", &a, &opr, &b);

    switch (opr) {
        case '+':
            printf("%f\n", a + b);
            break;
        case '-':
            printf("%f\n", a - b);
            break;
        case '*':
            printf("%f\n", a * b);
            break;
        case '/':
            printf("%f\n", a / b);
    }

    return 0;
}
```

We first initialize the variables `a` and `b` as `float` types (because the inputs are real numbers), and initialize the operator variable `opr` as a `char` type. Then, we scan the user input for the numbers and the operator. After that, we can return the result according to the operator from the user.

The compiled program works as follows:



```
max ex3-2 % ./operator
Enter operation (separate numbers and operators with a single space): 8 * 7
56.000000
max ex3-2 % ./operator
Enter operation (separate numbers and operators with a single space): 4.5 / 0.5
9.000000
max ex3-2 % ./operator
Enter operation (separate numbers and operators with a single space): 4.7 + 7.6
12.299999
max ex3-2 %
```