C FUNDAMENTALS PRACTICE - CONSTANTS

Table of Contents

1. READ README

- This file is a practice file for C fundamentals.
- When you're done with a section move the cursor on the section heading and type `S-<right>` (or SHIFT+<right-arrow>).
- When you leave class without having completed the file, save a copy to GDrive as a backup and/or to work on it from home
- When you've completed the file, upload it to Canvas

2. DONE Identify yourself

- replace `[yourName]` in the header of this file by your name
- add `(pledged)` next to your name (as in "I obey the honor code")
- Change the "TODO" in the headline to "DONE" (`S-<right>`)
- save the file (`C-x C-s`).

3. DONE Constants

1. Create a C code block named 1 with three different constant definitions.
2. Define the Arkansas sales tax rate (6.5%) as `SALES_TAX_AR` using the `#define` pre-processor macro.
3. Define the Euler number using `M_E` in `math.h`, and call it `EULER`.
4. Define the speed of light as `SPEED_OF_LIGHT` using `const`.

5. Print all three definitions to get the output:

```
The Euler number is: e = 2.7182818285
The AR sales tax is: 6.5%
The speed of light is: 299792458 m/s
```

6. Tip: the `%` character is reserved for format specification. To escape it (i.e. to print `%`, use `%%` in the `printf` statement).
7. Tip: You only need to include extra libraries - `stdio.h` is already included, and `main` will be added automatically.

—— PUT CODE BLOCK BELOW THIS LINE ——

```
// include libraries
#include <math.h>

// define constants
#define SALES_TAX_AR 6.5f
#define EULER M_E
const int SPEED_OF_LIGHT=299792458;

// main program
printf("The Euler number is: e = %.10f\n", EULER);
printf("The AR sales tax is: %.1f%%\n", SALES_TAX_AR);
printf("The speed of light is: %d m/s\n", SPEED_OF_LIGHT);
```

```
The Euler number is: e = 2.7182818285
The AR sales tax is: 6.5%
The speed of light is: 299792458 m/s
```

```
The Euler number is: e = 2.7182818285
The AR sales tax is: 6.5%
The speed of light is: 299792458 m/s
```

4. DONE Standard math library

- Open the file math.h (from GitHub) and search for the definition of `M_PI`. What is the last non-zero digit? What's the precision?
- Answer: 6

- Can you also print it without using `math.h`? Write a quick 2-line program (in a C code block) to print this value![1]

```
# define pi "3.14159265358979323846"    /* M_PI */
printf("%s\n",pi);
```

```
3.14159265358979323846
```

5. DONE Reading input

1. Copy the code block 1 below into a code block 1
2. Modify 1 so that it reads a floating-point variable `x` instead of an integer variable `i`.
3. The *format specifier* for `float` numbers is `%f`.
4. Create an input file named `finput` in `$PWD` and put the number `3.141593` into it.

5. Run 1. You should get the result:

```
: Enter a floating-point number!
: You entered 3.141593
```

```
int i;
puts("Enter an integer!");
```

```
scanf("%d", &i);
printf("You entered %d\n", i);
```

```
Enter an integer!
You entered 5
```

```
float x;
puts("Enter a floating-point number!");
scanf("%f", &x);
printf("You entered %f\n", x);
```

6. DONE Naming identifiers

Naming conventions dictate that you should use

- upper case letters for constants
- lower case letters for variables and function names
- separate names with underscore or insert capital letters
- name according to function
- In the code block 1, complete the code according to these rules.
- Run the code block with the additional header-argument `:flags -Wall` to see if you get any warnings.

```
// integer constant for the speed of light
const int SPEED_OF_LIGHT = 299792458;

// floating-point constant for pi
#define PI 3.141593f

// integer variable for volume computations
int volume;

// character variable for last names
char lastName;

// function that adds two integers i and j
int add(int i,int j) {
  return i + j;
}

// variable whose name contains "my", "next", and "birthday"
int my_next_birthday;
```

7. DONE Fix the program

The program statements in 1 contain multiple errors. Find them all and fix them if you can so that the program compiles and runs without errors - without simply commenting out erroneous code.

```
int _void = 1;

double times10;

float _long = 10.45;

char else;
```

```
const int ui-1 = 1;

int bottles100 = 100;
```

8. DONE Program layout

The program 1 below does not accommodate program layout conventions (though it will compile and run). Fix that.

Tip: sort the different parts of the program first. The comments might be helpful for that.

Remember that <TAB> will correct indentation in the code block.

The output looks like this:

```
I'm gonna print a number now.
The number is 100
100*(-1)=-100
```

```
// declarations
const int X=100;
int y;

// print constant
puts("I'm gonna print a number now.");
printf("The number is %d\n", X);

// computation
int i=-1;
y = X * i;

// print result of computation
printf("%d*(%d)=%d\n",X,i,y);
```

```
I'm gonna print a number now.
The number is 100
100*(-1)=-100
```

9. DONE Fix the program

The program 1 violates layout standards and will not compile. Fix it and run it - the correct output is: `1 is not 2`.

```
#define ONE 1
#define TWO 2
printf("%d is not %d\n", ONE, TWO);
```

```
1 is not 2
```

Footnotes:

[1] The specific number you're seeing when trying to print the number `M_PI` from the math library `math.h`, 3.14159265358979311600, is the closest representation of π (pi) that can be achieved with a 64-bit double. When you ask printf to display the number with a precision of 20 decimal places, it fills in with additional digits beyond the accurate representation, which in this case do not match the true decimal expansion of π.

Author: yourname (pledged)

Created: 2024-02-17 Sat 09:12

Validate