Table of Contents

1 Volume

1.1 Problem

- Write a program that computes the volume of a sphere with a 10-meter radius, using the formula $v = 4/3 \times \pi r^3$.
- Write the fraction $4/3$ as `4.0f/3.0f`. (Try writing it as 4/3 and see what happens.)
- Remember that C does not have an exponentiation operator, so you need to write r^3 as `r*r*r`.
- Upload your solution program as a `.c` file or as a `.org` file to Schoology not later than 11 AM on Monday, February 21st. Make sure that your program actually runs without errors!
- Be prepared to present your solution in class.

1.2 Solutions and pseudo solutions

- In the first solution, we define the constant as a macro using `#define`.

```
#define PI_CONST 3.141593 // macro declaration

float volume, radius=10.0f; // type declaration

volume = 4.0f/3.0f * PI_CONST * radius * radius * radius; // statement

printf("The volume of a sphere of radius %.2f is %.2f\n",
       radius, volume); // display output
```

- What happens when you use `4/3` instead of `4.0f/3.0f` in the formula?

```
#define PI_CONST 3.141593 // macro declaration

float volume, radius=10.0f; // type declaration

volume = 4/3 * PI_CONST * radius * radius * radius; // statement

printf("The volume of a sphere of radius %.2f is %.2f\n",
       radius, volume); // display output
```

```
The volume of a sphere of radius 10.00 is 3141.59
```

- What happens when you use integers instead of floating point numbers?

```
#define PI_CONST 3.141593 // macro declaration

int volume, radius=10; // type declaration

volume = 4.0f/3.0f * PI_CONST * radius * radius * radius; // statement
```

```
printf("The volume of a sphere of radius %d is %d\n",
       radius, volume); // display output
```

- What happens when you use a less accurate PI constant?

```
#define PI_CONST 3.14 // macro declaration

int volume, radius=10; // type declaration

volume = 4.0f/3.0f * PI_CONST * radius * radius * radius; // statement

printf("The volume of a sphere of radius %d is %d\n",
       radius, volume); // display output
```

- In the second solution, we include the constant using the math library `math.h`.

```
#include <math.h>  // include math library with PI=M_PI

float volume, radius=10.0f; // type declaration

volume = 4.0f/3.0f * M_PI * radius * radius * radius; // statement

printf("The volume of a sphere of radius %.2f is %.2f\n",
       radius, volume); // display output
```

1.3 Reading input

- For the expanded problem, we use the input function `scanf` for the `radius`. Unfortunately, Emacs Org-mode will not wait for us to enter the input. You have to tangle the code block, compile and run the C file outside of Emacs, on the CMD shell (terminal).

- In the code below, I'm using a trick - the Org-mode header argument `cmdline` will accept input from a file. This file, `in.txt`, contains my "interactive" input - the number `10`

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

float volume, radius; // declare types

printf("Enter radius of sphere: \n");  // ask for input
scanf("%f", &radius);                  // read input

volume = (4.0f / 3.0f) * M_PI * radius * radius * radius; // statement

printf("Volume (cubic meters): %.1f\n", volume); // display result
```

- This does not seem to work.

```
#include <math.h> // include math library
#include <stdlib.h> // include standard library

float volume, radius; // declare types

printf("Enter radius of sphere: \n");  // ask for input
//       scanf("%f", &USER_INPUT);                  // read input
radius = atof(USER_INPUT);
volume = (4.0f / 3.0f) * M_PI * radius * radius * radius; // statement

printf("Volume (cubic meters): %.1f\n", volume); // display result
```

- To pass arguments to source code blocks, use `var` (cp. manual).

```
int radius = INPUT;
printf("Input: %d", radius);
```

- We can use this for a better solution: this code block can be run again and again like the compiled program, and we enter the variable `radius` as header `:var RADIUS`.

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

float volume, radius; // declare types

radius = RADIUS;

volume = (4.0f / 3.0f) * M_PI * radius * radius * radius; // statement

printf("Volume (cubic meters): %.1f\n", volume); // display result
```

- See also:
  - stackexchange (2018)
  - stackexchange (2015)
  - stackoverflow (2017)

1.4 Review (class notes)

1.4.1 Examples - can you see how these could be improved?

- Exhibit 1

```c
#include <stdio.h>

int main(void) {

  int radius, volume;
  radius = 10;
  volume = 4.0f/3.0f * 3.14 * radius * radius * radius;

  printf("Dimensions: %d\n", radius);
  printf("Volume (cubic meters): %d\n", volume);

  return 0;
}
```

Dimensions: 10 Volume (cubic meters): 4186

- Exhibit 2

```
#define PI 3.14;
int main()
{
  int r = 10;
  int v = 4.0f/3.0f * 3.141592 * r*r*r;

  printf("volume = %d",v);
}
```

```
volume = 4188
```

Author: Marcus Birkenkrahe
Created: 2022-03-17 Thu 17:02
Validate