

C Programming Exercise Solution for Beginner

July 10, 2025

Abstract

i'm currently learning C and planning to improve my \LaTeX skill. so i came up with this idea. This is only for my learning purpose. doing C problem while writing the solution and my approach/solution to a \LaTeX Note. i'm gonna explain my approach to solve given problem.

1. Calculate product of two integers

Write a C program that accepts two integers from the user and calculates the product of the two integers. Test Data :

```
> Input the first integer: 25
> Input the second integer: 15
Expected Output:
> Product of the above two integers = 375
```

1.1 Solution

```
1  #include <stdio.h>
2
3  #define HEIGHT 7
4  #define WIDTH 5
5  #define RADIUS 6
6  #define M_PI 3.14159265358979323846
7
8  // global scope variable declaration
9  int num1, num2, result;
10
```

Listing 1: Defining required Variable

when you declaring a number using a `#define`, It has no type. It is a simple text substitution. although `PI` is already defined by C itself, but i found on a stack overflow thread, they were defining `PI` with `#define`. this kind of problem can be tackled easily by creating a separate *function* outside of `main()` function, and declaring those variable using a *global scope variable* for efficiency and avoiding conflict within other function. while defining a global scope variable, i'm creating a function to calculate the integers:

```
1  void CalculateTwoIntegers(int num1, int num2){
2      result = num1 + num2;
3      printf("The result is: %d\n", result);
4  }
5
```

Listing 2: function

1.2 Pros And Cons

by storing a number in `#define`, it's not the best practice in the modern C. because as i said, **It has no data types, hard to debug, there's no scope, and etc.** i'd recommend using a `const` variable instead. for example: `const double PI = 3.1415926535;`. a `#define` is more suitable for storing a macro rather than a number for mathematical operation unless you know what are you doing.

2. Calculate average weight for purchases

Write a C program that accepts two item's weight and number of purchases (floating point values) and calculates their average value. for example:

```

Weight - Item1: 15
No. of item1: 5
Weight - Item2: 25
No. of item2: 4
Expected Output:
Average Value = 19.444444

```

2.1 Solution

by using math shit for this problem, i've figured out using average value for calculating the result, we can implement this solution to our code:

$$\text{Average Value} = \frac{(w_1 \times n_1) + (w_2 \times n_2)}{n_1 + n_2} \quad (1)$$

As shown in Equation 1, By creating a separate function and declaring the required variable first using `double` data types, finally we can implement the math formula above to our code solution.

```

1 void CalculatePurchases(){
2     double wi1, ci1, wi2, ci2, resultDouble;
3
4     printf("Enter Weight for item 1: ");
5     scanf("%lf", &wi1);
6
7     printf("Enter Weight for ci1: ");
8     scanf("%lf", &ci1);
9
10    printf("Enter Weight for weight item 2: ");
11    scanf("%lf", &wi2);
12
13    printf("Enter Weight for ci2: ");
14    scanf("%lf", &ci2);
15
16    resultDouble = ((wi1 * ci1) + (wi2 * ci2)) / (ci1 + ci2);
17    printf("The result is %lf\n", resultDouble);
18 }
19

```

Listing 3: Code Solution

Why does `scanf()` require to use `&` ? well, It needs to change the variable. Since all arguments in C are passed by value you need to pass a pointer if you want a function to be able to change a parameter. Here's a super-simple example showing it:

```

1 void nochange(int var) {
2     // Here, var is a copy of the original number. &var != &value
3     var = 1337;
4 }
5 void change(int *var) {
6     // Here, var is a pointer to the original number. var == &value
7     // Writing to `*var` modifies the variable the pointer points to
8     *var = 1337;
9 }
10 int main() {
11     int value = 42;
12     nochange(value);
13     change(&value);
14     return 0;
15 }
16

```

Listing 4: scanf example

C function parameters are always "pass-by-value", which means that the function `scanf` only sees a copy of the current value of whatever you specify as the argument expression. In this case `&i` is a pointer value that refers to the variable `i`. `scanf` can use this to modify `i`. If you passed `i`, then it would only see an uninitialized value, which (a) is UB, (b) is not sufficient information for `scanf` to know how to modify `i`.

3. Print employee ID and monthly salary

Write a C program that accepts an employee's ID, total worked hours in a month and the amount he received per hour. Print the ID and salary (with two decimal places) of the employee for a particular month. Test Data :

```

Input the Employees ID(Max. 10 chars): 0342
Input the working hrs: 8
Salary amount/hr: 15000
> Expected Output:
Employees ID = 0342
Salary = U\$ 120000.00

```

Listing 5: Input

for this problem, we need three type of data:

1. ID, it can be `char`, `int`, or an `array`. depends on the problem.
2. working hours, with `int` data types.
3. Salary amount, `float` or `double` is fine.

by using this math formula to calculate the result:

$$Salary = hoursworked \times rateperhour \quad (2)$$

3.1 Solution

Thus, as shown in equation 2, we can implement this math formulae to our code. in my opinion, my approach is too common but i believe this is an efficient way. and more importantly, i don't care about your opinion.

```

1  int working_hours, id;
2  double salary_amount_per_hour;
3
4  float CalculateWorkingHours()
5  {
6      printf("Enter employees ID: ");
7      scanf("%d", &id);
8
9      printf("Enter working hours: ");
10     scanf("%d", &working_hours);
11
12     printf("Enter salary_amount per hours: ");
13     scanf("%lf", &salary_amount_per_hour);
14
15     return working_hours * salary_amount_per_hour;
16 }
17

```

Listing 6: Code Implementation

by creating a separate function, i'm using `float` to return a float data types when the function is called. if you want to return a more accurate data types i'd recommend using `double` types. but in this case we're not performing a complex math operation so `float` is fine for me. Thus, i called the function and store it as a variable in order to print the result.

```

1  int main()
2  {
3      float salary = CalculateWorkingHours();
4      printf("Salary = U\$ %.2f\n", salary);
5      return EXIT_SUCCESS;
6  }
7

```

Listing 7: Call the function