22GES1698
J. NIRUSH RUBAN

Assignment 02: User-Defined Functions in C Language
FC11322

References : Problem solving and
Program design in C by Jeri R. Hanly & Elliot B. Koffman.

Part 1: Theory

1. What is a function in C?
   - Functions are coding modules used by programmers to facilitate code reuse.
2. What are the differences between library functions and user-defined functions?
   - Library functions are pre-written functions given by the c standard library while user defined functions are the functions created by the programmer for a specific purpose.
3. What are the advantages of using functions in a program?
   - Modularity by breaking code into smaller segments
   - Enables code reusability as functions can be used limitless times
   - Better maintenance of the code as testing and debugging is made easier.

4. What is the basic syntax for declaring and defining a function in C?
   - return_type function_name(parameter_list) { // Function body }
5. What is the purpose of the return statement in a function?
   - It is to pass the value from a function to the call function
6. What does a void return type signify?
   - The void return type in C signifies that a function does not return any value to the calling function. Instead, such a function is used to perform an action or a sequence of actions without providing a result.

7. What are the differences between formal parameters and actual parameters?
   - Formal parameter is an identifier that represents a corresponding actual argument in a function definition. They exist only inside the function where they are declared.
   - Actual parameters are the real values passed to the functions when it is called. They exist in the calling function and they are used to initialize formal parameters during function call.
8. What is the difference between call by value and call by reference?
   - In call by value, of the actual argument is passed to the function. The function works with the copy, and changes made to the parameter inside the function do not affect the original argument.
   - In call by reference, the memory address of the actual argument is passed to the function. The function operates directly on the original argument, so changes made to the parameter inside the function affect the original argument.
9. Can a function have no parameters? Provide an example.
   - Yes. Some functions do not require any input values to perform their tasks and they rely on variables declared within them or global data. For example, void greet(){ printf(“helloo world\n”);} function does not require any input parameters. Mostly these kind of functions are used to display messages or initialize resources.

Scope of Variables

10. What is the scope of a variable in C?
    - The scope of a variable is the portion of the program where the variable can be used or accessed. This scope can be local, global or functional.
11. How does the scope of a variable affect function behavior?
    - Variables with local scope are only accessible within the function and any change within the fuction does not affect the main program.
    - Variables with global scope are accessible within any function and hence any change made to these variables can affect other parts of the program. This can lead to unfavourable effects if the varialbles are unintentionally modified.
12. What are local and global variables? Provide examples.
    - Local variables: variables declared within a function or a block that are accessible only within the function.
    - E+g:
      ```
      #include <stdio.h>
      void display() {
      int localVar = 5; // Local variable
       printf("Local variable = %d\n", localVar); }
      int main() {
      display();
      return 0; }
      ```
    - Global variables: variables declared outside of all functions usually at the top of all functions.

    -
      ```
      Int gvar = 10;
      Void modifygvar(){
      gvar += 5;
      printf(“%d\n”, gvar);
      }
      Int main(){
      Modifygvar();
      return 0;
      }
      ```

Function Types

13. What are the four categories of functions in C?

    - No arguments, no return value.
    - Arguments, but no return value.
    - No arguments, but a return value.
    - Both arguments and a return value

Provide examples of functions with:

14. No arguments, no return value

```c
void displayMessage() { // No arguments, no return value
    printf("Welcome to the program!\n");
}

int main() {
    displayMessage();
    return 0;
}
```

15. Arguments, but no return value

```c
void greet(char name[]) { // Takes an argument but does not return a value
    printf("Hello, %s!\n", name);
}

int main() {
    greet("Alice");
    return 0;
}
```

16. No arguments, but a return value

```c
int getNumber() { // No arguments, but returns a value
    return 42;
}

int main() {
    int num = getNumber();
    printf("Number = %d\n", num);
    return 0;
}
```

17. Both arguments and a return value

```c
int add(int a, int b) { // Takes two arguments and returns a value
    return a + b;
}

int main() {
    int result = add(5, 10);
    printf("Sum = %d\n", result);
    return 0;
}
```

Function Prototypes

18. What is a function prototype?
    - It is a declaration of the function with its name, parameters and return type without the actual body of the function. It tells the compiler what to expect when the function is called.
19. Is it mandatory to use a function prototype? Why or why not?
    - Not mandatory if a function is defined before it is called as the function already knows the function's signature. If functions are called before their definition then a prototype is required.

Error Handling in Functions

20. What happens if a function does not include a return statement for a non-void return type?
    - If a function with a non viod return type does not include a return statement then the behaviour is undefined. This will result in unpredictable behaviour and runtime errors.
21. Can a function have multiple return statements? Provide an example.
    - Yes, a function in c can have multiple return statements but only one of them will execute during a single function call.
    - E+g: const char* checknum(int num){
      If (num > 0){
      Return "pos";}
      Else{
      Return "neg";}
      }

Part 2: Practical

1. Write a function to print a greeting message, such as "Hello, welcome to C programming!"

```
void greet(){
    printf("hello, welcome to c programming\n");
}
```

2. Write a function to add two integers and return the result.

```
int add2num(int x, int y){
    return x + y;
}
```

3. Write a function to calculate and return the square of a given number.

```
int calcsquare(int n){
    return n * n;
}
```

4. Write a function to check whether a number is even or odd.

```
void checknum(int n){
    if(n % 2 == 0){
        printf("even number\n");
    }else{
        printf("odd number\n");
    }
}
```

5. Write a function to find and return the largest of two given numbers.

```
int compare(int a, int b) {
return (a > b) ? a : b;
}
```