

TUTORIAL 6

1. Explain why function prototype is required?

The function prototypes are used to tell the compiler about the number of arguments and about the required datatypes of a function parameter, it also tells about the return type of the function.

2. Mention the reason why scope is important?

Scope defines where variables can be accessed or referenced.

3. If the global variables can be used anywhere in the program, why not make all global variables global?

So, by using a local variable you decrease the dependencies between your components, i.e. you decrease the complexity of your code. You should only use global variable when you really need to share data, but each variable should always be visible in the smallest scope possible.

4. State the advantage of using register storage class? What are the functions with register storage class?

That means it will be stored in the memory. The main advantage of storing the variable as register is they are stored in CPU memory which is accessed very fast compared to RAM. This makes the program to get executed faster. Hence these type of variables are mainly used where quick access to them are required.

5. What is a linkage?

linkage describes how names can or can not refer to the same entity throughout the whole program or one single translation unit.

6. Explain the linkage intend? State the different types of linkages?

- If an identifier has file scope and does not use the static storage class specifier at the time of the first declaration, the identifier has the external linkage. The externally linked identifier or function visible to all translation unit of the program that means we can access it any translation unit of the program.
- If a global identifier declares with static storage class, its linkage will be internal. An identifier implementing internal linkage is not accessible outside the translation unit in which it is declared. An identifier with internal linkage denotes the same object or function within one translation unit if it is accessed by any function.
- None linkage- A local variable has no linkage and refers to unique entities. If an identifier has the same name in another scope, they do not refer to the same object.

7. Differentiate between internal static and external static variable?

Internal static variables are similar to auto(local) variables.

External static variables are similar to global(external) variables.

Internal static variables are active(visibility) in the particular function.

External Static variables are active(visibility)throughout the entire program.

Internal static variables are alive(lifetime) until the end of the function.

External static variables are alive(lifetime) in the entire program.

8. What does extern mean in a function declaration?

extern specifies that the variable or function is defined in another translation unit.

9. Compare recursion and iteration?

Recursion

- Recursion uses selection structure.
- Infinite recursion occurs if the recursion step does not reduce the problem in a manner that converges on some condition (base case) and Infinite recursion can crash the system.
- Recursion terminates when a base case is recognized.
- Recursion is usually slower than iteration due to the overhead of maintaining the stack.
- Recursion uses more memory than iteration.
- Recursion makes the code smaller.

Iteration

- Iteration uses repetition structure.
- An infinite loop occurs with iteration if the loop condition test never becomes false and Infinite looping uses CPU cycles repeatedly.
- An iteration terminates when the loop condition fails.
- An iteration does not use the stack so it's faster than recursion.
- Iteration consumes less memory.
- Iteration makes the code longer.

10. Can main() called recursively?

The main function is just like any other function. Just like other functions, you can call main from anywhere you want. So, yes you can recursively call main.

11.Explain the keywords: a) Actual parameters b) formal parameters c) space complexity d) time complexity e) storage class specifiers

a) The arguments that are passed in a function call are called actual arguments

b) These are the variables or expressions referenced in the parameter list of a subprogram specification. The datatype of the receiving value must be defined. The scope of formal arguments is local to the function definition in which they are used.

c) The space complexity of an algorithm or a computer program is the amount of memory space required to solve an instance of the computational problem as a function of characteristics of the input. It is the memory required by an algorithm until it executes completely.

d) the time complexity is the amount of time taken by an algorithm to run, as a function of the length of the input

e) A storage class specifier is used to refine the declaration of a variable, a function, and parameters. Storage classes determine whether: The object has internal, external, or no linkage.

12. What will be the output of the below program?

```
int a =10;
```

```
void compute( int a)
```

```
{
```

```
    a =a;
```

```
}
```

```
int main ()
```

```
{
```

```
    int a =100;
```

```
    printf(“%d”, a);
```

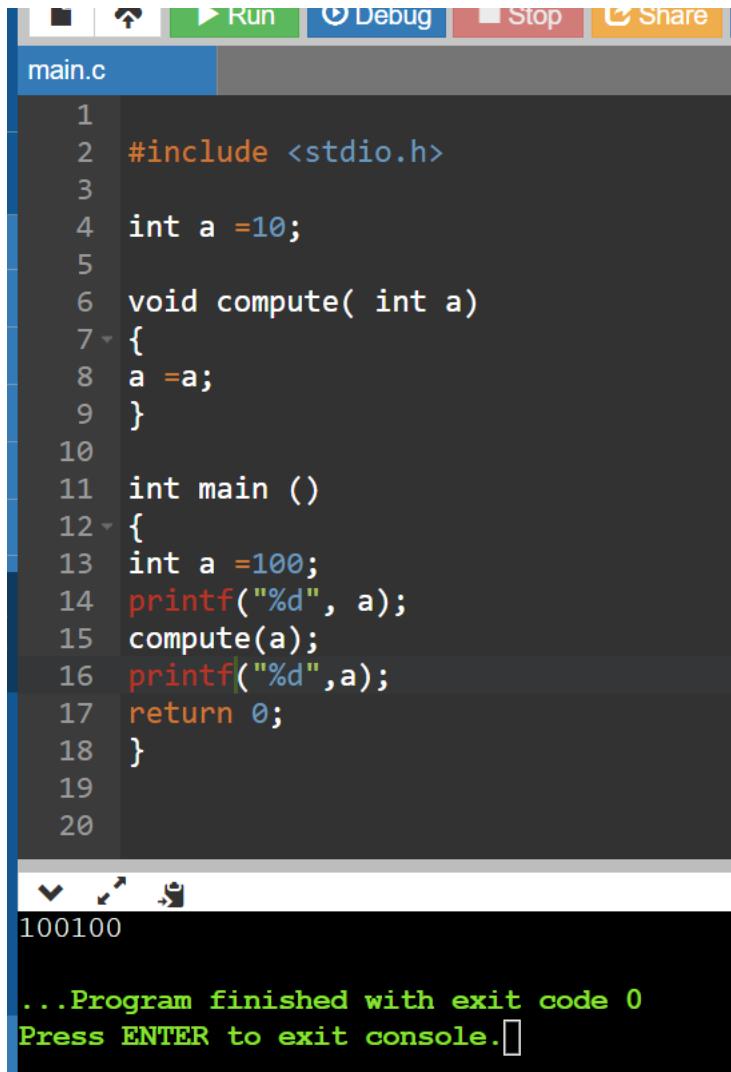
```
compute(a);
```

```
printf("%d",a);
```

```
return 0;
```

```
}
```

Output will be 100100

A screenshot of a code editor window showing a C program. The editor has a dark theme with a toolbar at the top containing icons for Run, Debug, Stop, and Share. The file name 'main.c' is visible in the top left. The code is as follows:

```
1
2  #include <stdio.h>
3
4  int a =10;
5
6  void compute( int a)
7  {
8      a =a;
9  }
10
11 int main ()
12 {
13     int a =100;
14     printf("%d", a);
15     compute(a);
16     printf("%d",a);
17     return 0;
18 }
19
20
```

Below the code editor, there is a console window showing the output '100100' and the message '...Program finished with exit code 0 Press ENTER to exit console.'.

Practice!!

Write a menu-based program in C that uses the set of functions to perform the following

operations

(a) Reading a complex number

(b) Writing a complex number

(c) Addition of two complex number

(d) Subtraction of two complex number

(e) Multiplication of complex number

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
struct complex
```

```
{
```

```
    int real, img;
```

```
};
```

```
int main()
```

```
{
```

```
    int choice, x, y, z;
```

```
    struct complex a, b, c;
```

```
    while(1)
```

```
    {
```

```
        printf("Press 1 to add two complex numbers.\n");
```

```
        printf("Press 2 to subtract two complex numbers.\n");
```

```
        printf("Press 3 to multiply two complex numbers.\n");
```

```
        printf("Press 4 to divide two complex numbers.\n");
```

```
        printf("Press 5 to exit.\n");
```

```

printf("Enter your choice\n");
scanf("%d", &choice);

if (choice == 5)
    exit(0);

if (choice >= 1 && choice <= 4)
{
    printf("Enter a and b where a + ib is the first complex number.");
    printf("\na = ");
    scanf("%d", &a.real);
    printf("b = ");
    scanf("%d", &a.img);
    printf("Enter c and d where c + id is the second complex
number.");
    printf("\nc = ");
    scanf("%d", &b.real);
    printf("d = ");
    scanf("%d", &b.img);
}
if (choice == 1)
{
    c.real = a.real + b.real;
    c.img = a.img + b.img;
}

```

```

    if (c.img >= 0)
        printf("Sum of the complex numbers = %d + %di", c.real, c.img);
    else
        printf("Sum of the complex numbers = %d %di", c.real, c.img);
}

else if (choice == 2)
{
    c.real = a.real - b.real;
    c.img = a.img - b.img;

    if (c.img >= 0)
        printf("Difference of the complex numbers = %d + %di", c.real,
c.img);
    else
        printf("Difference of the complex numbers = %d %di", c.real,
c.img);
}

else if (choice == 3)
{
    c.real = a.real*b.real - a.img*b.img;
    c.img = a.img*b.real + a.real*b.img;

    if (c.img >= 0)

```



```

    printf("Multiplication of the complex numbers = %d + %di",
c.real, c.img);
    else
        printf("Multiplication of the complex numbers = %d %di", c.real,
c.img);
    }
    else if (choice == 4)
    {
        if (b.real == 0 && b.img == 0)
            printf("Division by 0 + 0i isn't allowed.");
        else
        {
            x = a.real*b.real + a.img*b.img;
            y = a.img*b.real - a.real*b.img;
            z = b.real*b.real + b.img*b.img;

            if (x%z == 0 && y%z == 0)
            {
                if (y/z >= 0)
                    printf("Division of the complex numbers = %d + %di", x/z, y/z);
                else
                    printf("Division of the complex numbers = %d %di", x/z, y/z);
            }
            else if (x%z == 0 && y%z != 0)

```

```

{
    if (y/z >= 0)
        printf("Division of two complex numbers = %d + %d/%di", x/z,
y, z);
    else
        printf("Division of two complex numbers = %d %d/%di", x/z, y,
z);
}
else if (x%z != 0 && y%z == 0)
{
    if (y/z >= 0)
        printf("Division of two complex numbers = %d/%d + %di", x, z,
y/z);
    else
        printf("Division of two complex numbers = %d %d/%di", x, z,
y/z);
}
else
{
    if (y/z >= 0)
        printf("Division of two complex numbers = %d/%d +
%d/%di",x, z, y, z);
    else
        printf("Division of two complex numbers = %d/%d %d/%di", x,
z, y, z);
}

```

```
    }  
    }  
}  
else  
    printf("Invalid choice.");  
  
    printf("\nPress any key to enter choice again...\n");  
}  
return 0;  
}
```