Certificate - Software Development

Functions, Data Scope & Pointers

Functions in C

A function is a group of statements that is executed, when it is called from some point of the program. We can structure our programs in a more modular way using them.

Syntax of a Function

```
retrun_type function_name( para1, par2, …)
{
    Statement_1;
    Statement_2;
}
```

- **return_type** – is the data type specifier of the value returned by the function.
- **function_name** – is the identifier used to call the function.
- **para1.. paraN** – Parameters are the data received by it.

A Function - Example

```
# include <stdio.h>
main()
{
    int a,b,sum;
    printf("Enter 2 integers : ");
    scanf("%d %d",&a,&b);
    sum = addition(a,b); // Calling the function
    printf("Sum = %d\n",sum);
    /* 2nd call directly prints the return value */
    printf("New Sum = %d\n",addition(10,33));
}

int addition(int x, int y)
{
    int temp;
    temp = x+y;
    return temp;
}
```

Output

```
Enter 2 integers : 12 75
Sum = 87
New Sum = 43
```

Scope of Data

- With the introduction of functions we need to explain about the scope of data in C language.
- Scope of a data item defines the accessibility range or limit of the data item. In C language we can give 3 different scope levels for data.

1. **Global Scope**
   - These are data that can be accessed from any function in the program.
   - However, if any function declares a local variable by the same name as a global variable then that function is unable to access the particular global data item. This is due to a visibility issue rather and not due change of scope of the global data item.
   - ***All global data items should be defined before the main() function.***

Scope of Data Ctd..

2. Local Scope

- A data item declared inside a function will have a local scope.
- That is they can be accessed only within the function where they are declared.
- All parameters passed into a function also will also have a local scope.

3. Partially Global

- These are data items that can be accessed by a subset of functions but not by all of the functions within the program.
- In C language they are the ***data item that are declared in between two functions*** as they can only be accessed by a function which follows the declaration.

Data Scoping - Example

```
    int global_X = 10;
main()
{   printf("IN MAIN\n");
    int local_A =20;
    printf("Global X = %d\n",global_X);
    printf("Local A in main() = %d\n",local_A);
    sub();
}
     int part_global_Y = 30;

void sub()
{   printf("\nIN SUB\n");
    int local_B = 40;
    printf("Global X = %d\n",global_X);
    printf("Partially Global Y = %d\n",part_global_Y);
    printf("Local B in sub() =%d\n",local_B);
    newsub();
}
```

Output

```
IN MAIN
Global X = 10
Local A in main() = 20

IN SUB
Global X = 10
Partially Global Y = 30
Local B in sub() = 40
```

```
void newsub()
{
printf("\nIN NEWSUB\n");
int global_X = 500;
printf("Global X = %d\n…..
printf("Partially ….
}
```

Data Scoping - Example

```
int global_X = 10;
main()  .....
int part_global_Y = 30;
void sub()
{  printf("\nIN SUB\n");
   int local_B = 40;
   printf("Global X = %d\n",global_X);
   printf("Partially Global Y = %d\n",part_global_Y);
   printf("Local B in sub() =%d\n",local_B);
   newsub();
}
void newsub()
{
   printf("\nIN NEWSUB\n");
    int global_X = 500;
   printf("Global X = %d\n",global_X);
   printf("Partially Global Y = %d\n",part_global_Y);
}
```

Output

```
IN MAIN
Global X = 10
Local A in main() = 20

IN SUB
Global X = 10
Partially Global Y = 30
Local B in sub() = 40

IN NEWSUB
Global X = 500
Partially Global Y = 30
```

Procedures

- A module that doesn’t return a value back to the calling function is called a procedure.
- They too can have parameters just like functions. In C language there is no separate syntax for declaration of procedures.
- They are defined with the return type as “void” and then they cannot have a return keyword in the body of the function.

```
void newsub()
{
   printf("\nIN NEWSUB\n");
   int global_X = 500;
   printf("Global X = %d\n",global_X);
   printf("Partially Global Y = %d\n",part_global_Y);
   // ← No return keyword in the body
}
```

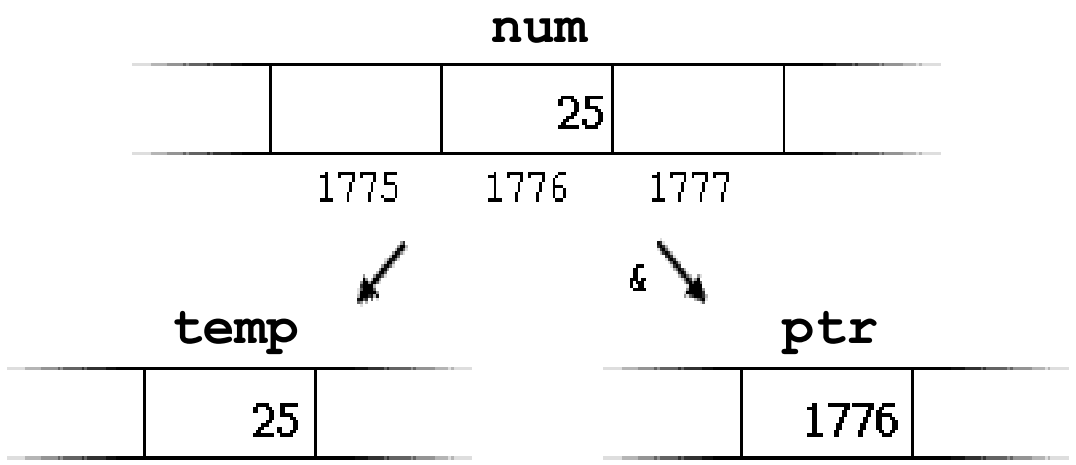
Pointers & References

- The memory of your computer can be imagined as a succession of memory cells(each cell is 1 byte). These cells are numbered in a consecutive way which is called the cell address (memory address).
- **Reference Operator(&)**

  The reference operator allow us to obtain address of a variable
- **Pointer (*) type**

  Is needed to store an address

Pointers Example
`char num = 25;`
`char temp = num;`
`char *ptr = &num;`

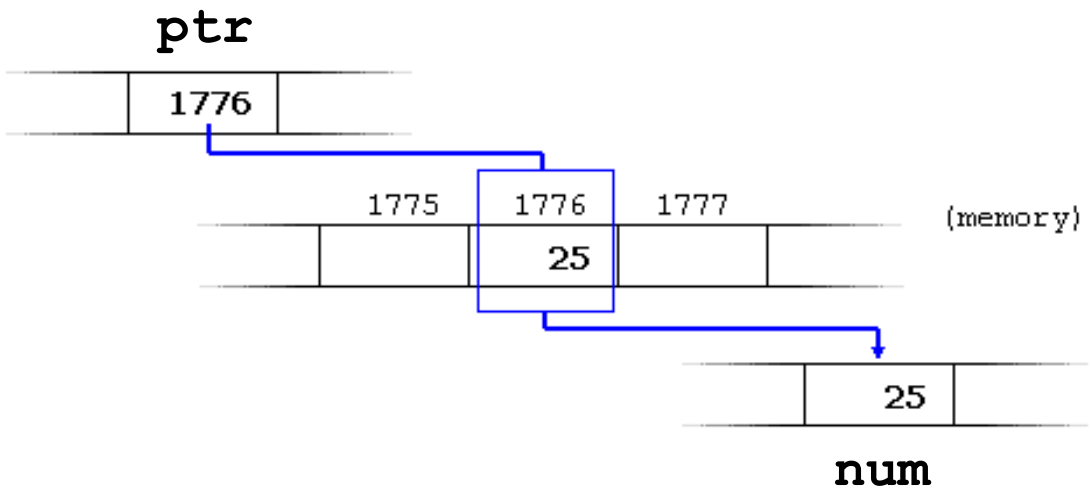


Pointers & References

- **Dereference Operator(*)**

  Using a pointer we can directly access the value stored in the variable which it points to. To do this, we simply have to precede the pointer's identifier with an asterisk (*), which acts as dereference operator and that can be literally translated to "**value pointed by**"

Pointers Example
`char num = 25;`
`char temp = num;`
`char *ptr = &num;`
`printf("Address is %d\n", ptr);`
`printf("Values is %d\n", *ptr);`

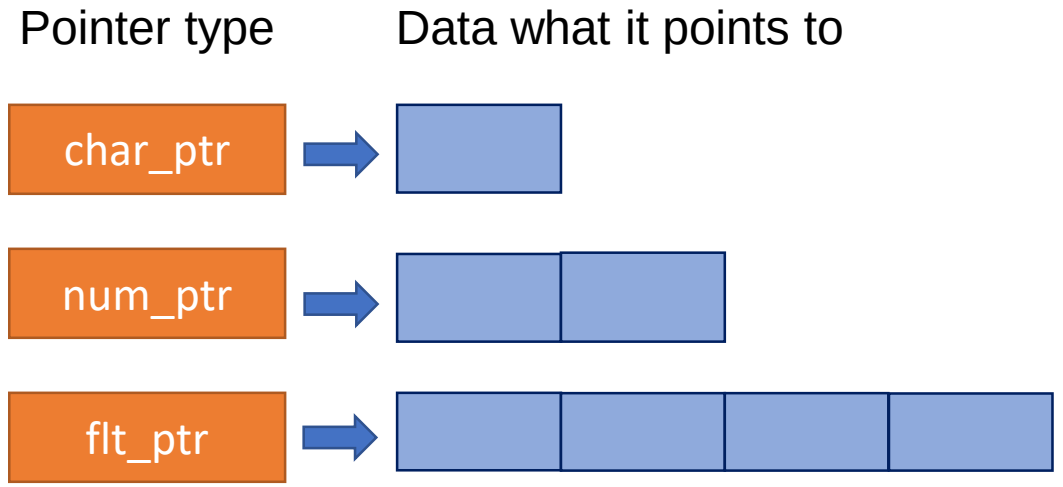


Declaring variables of pointer types

- The declaration of pointers follows this format:

Data_type *name;

Pointers Declarations
`int *num_ptr;`
`Char *char_ptr`
`float *flt_ptr;`



- These are three declarations of pointers. Each one is intended to point to a different data type, but in fact all of them are pointers and all of them will occupy the same amount of space in memory but the target that they point to is of different sizes.

Pointer Example

```
#include <stdio.h>
main ()
{
   int first = 10, second = 20;
   int *mypointer;
   printf("First Value = %d\n", first);
   printf("Second Value = %d\n", second);
   mypointer = &first;
   *mypointer = 50; // first is changed
   printf("First is changed to %d\n", first);
   mypointer = &second;
   printf("Now mypointer point to second : %d\n",*mypointer);
   printf("The actual content of mypointer : %d\n",mypointer);
   printf("Last output should be the address of Second\n");
}
```

Output
First Value = 10
Second Value = 20
First is changed to 50
Now mypointer point to second : 20
The actual content of mypointer : 6487568
Last output should be the address of Second

Lesson Summary

- Functions in C
- Data Scope
- Global, Local and Partially Global Data
- Data Scope Example
- Procedures
- Pointers and References
- Pointer Example