Short-term Hands-on Supplementary Course on C programming

Session 7:
Functions

Nandakishor. V
Nitheesh Kumar. N

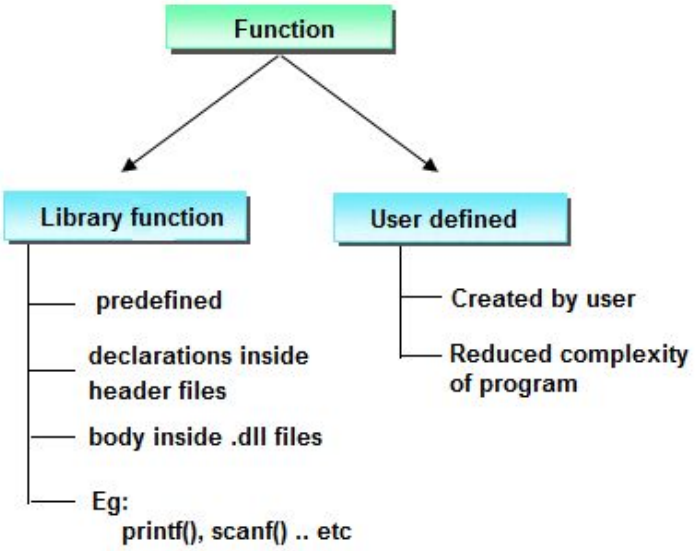
Agenda

1. What are functions?
2. Why do we need functions?
3. Functions in C
   a. Before main()
   b. Prototype for after nain()
   c. Macros
4. Parameter Passing: Pass by value and Pass by reference
5. Functions and Arrays
6. Const function parameters
7. Tutorial: Pass-by-Value and Pass-by-Reference

Functions

At times, we may need a certain portion of the code to be repeated. Instead of rewriting, it is better to write them as a subroutine which is called a function in C.

C enables its programmers to break up a program into segments commonly known as functions, each of which can be written more or less independently of the others.



Why do we need functions?

1. **Divide and Conquer / Modular Programming:** Dividing the program into separate well defined functions facilitates each function to be written and tested separately. This simplifies the process of getting the total program to work.
2. **Avoid repeating codes:** It is easy to copy and paste, but hard to maintain and synchronize all the code.
3. **Understandability:** Understanding, coding and testing multiple separate functions are far easier than doing the same for one huge function.
4. **Software Reuse:** you can reuse the functions in other programs, by packaging them into library codes.

Library functions

- The system provided these functions and stored them in the library. Therefore it is also called Library Functions. e.g. scanf(), printf(), strcpy, strlwr, strcmp, strlen, strcat, etc.
- You must include the appropriate C header files to use these functions.

Some commonly used libraries

Header File	Description
<ctype.h>	Character testing and conversion functions
<math.h>	Mathematical functions
<stdio.h>	Standard I/O functions
<stdlib.h>	Utility functions
<string.h>	String handling functions
<time.h>	Time manipulation functions

Some math lib functions

Function	Return Type	Use
ceil(d)	double	Returns a value rounded up to next higher integer
floor(d)	double	Returns a value rounded up to next lower integer
cos(d)	double	Returns the cosine of d
sin(d)	double	Returns the sine of d
tan(d)	double	Returns the tangent of d
exp(d)	double	Raise e to the power of d
fabs(d)	double	Returns the absolute value of d
pow(d1, d2)	double	Returns d1 raised to the power of d2
sqrt(d)	double	Returns the square root of d

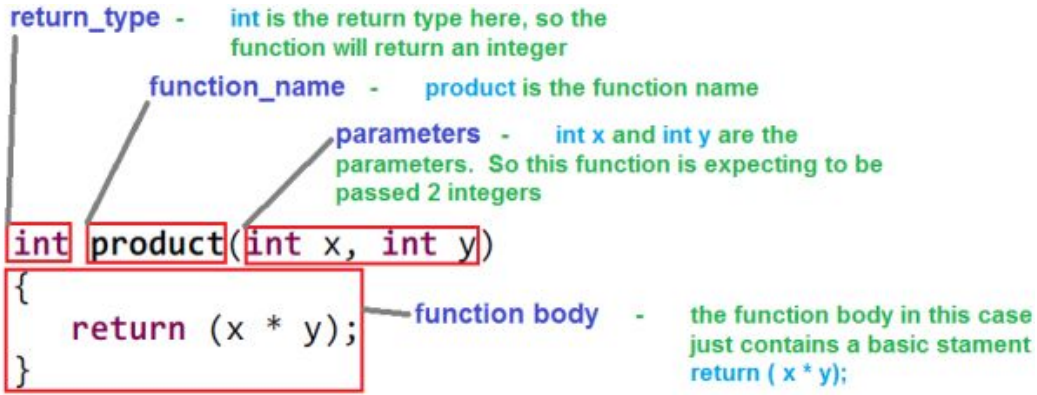
Different ways of defining

1. `int square( int x, int y) { }`

2. `int square(int , int );`
   `int square( int x, int y)`

3. `#define SQUARE(x) (x*x)`

4. `#include<math.h>`
   `pow(num,2)`



User defined functions - Declaration

```
1  int a = 10, b = 5, c;
2
3  int product(int x, int y);
4
5  int main(void)
6  {
7      c = product(a,b);
8
9      printf("%i\n",c);
10
11     return 0;
12 }
13
14 int product(int x, int y)
15 {
16     return (x * y);
17 }
```

Function Prototype - int is the return type and int x and int y are the function arguments

Main Function - int is always the return type and there are no arguments, hence the (void). Curly braces { } mark the start and end of the main function

Function call - product(a,b); a and b are global variables the function is passed. Here the values returned by the function are assigned to the variable c

Function Definition - contains the function statement return(x * y); the function returns x times y to the main function where it was called. Curly braces { } mark the start and end of the function

Actual and formal Parameters

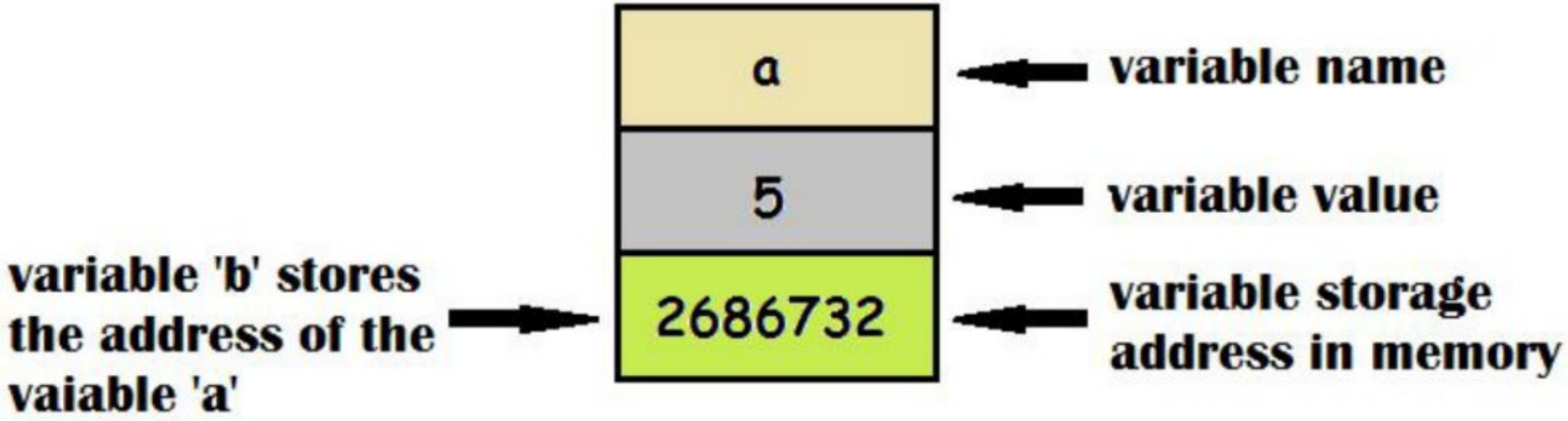
```
                    // Formal parameters
void add(int num1, int num2) // Function definition
{
    // Function body
}


int main()
{
            // Actual parameters
    add(10, 20); // Function call

    return 0;
}
```

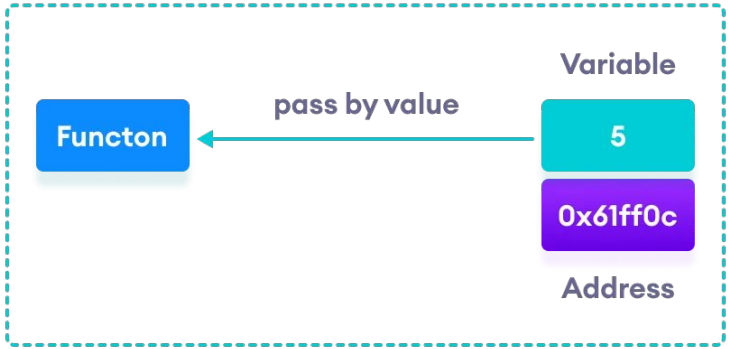
Variable storage in C

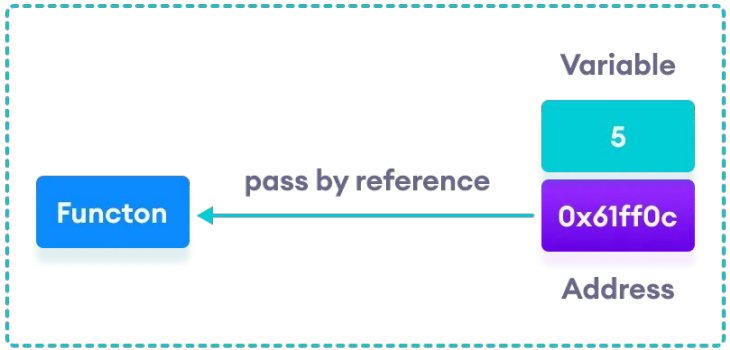
```
int a = 5;
int *b = &a;
```



Parameter Passing

- Pass by value
- Pass by reference

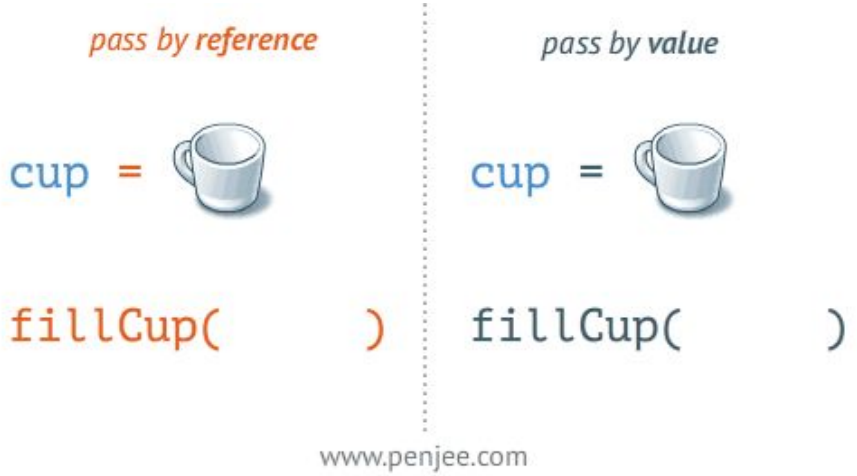




Let's see some examples

Pass-by-Value	Pass-by-Reference
`void swap(int a, int b)`	`void swap(int& a, int& b)`

Pass-by-Value Vs Pass-by-Reference



PASS BY VALUE	PASS BY REFERENCE
Mechanism of copying the function parameter value to another variable	Mechanism of passing the actual parameters to the function
Changes made inside the function are not reflected in the original value	Changes made inside the function are reflected in the original value
Makes a copy of the actual parameter	Address of the actual parameter passes to the function
Function gets a copy of the actual content	Function accesses the original variable's content
Requires more memory	Requires less memory
Requires more time as it involves copying values	Requires a less amount of time as there is no copying

Visit www.PEDIAA.com

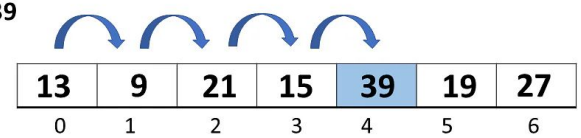
Functions and Arrays

```
int linearSearch(int arr[], int N, int x);
```

```
// search for element in an array
// returns the index of element if found else returns -1
int linearSearch(int arr[], int N, int x)
{
    for (int i = 0; i < N; i++)
        if (arr[i] == x)
            return i;
    return -1;
}
```

Searched Element

39



Tutorial

1. Write a C function which changes the value of the variable by reference.

```
value of num before function is 2
value of num in function is 12
value of num after function is 12
```

2. Write a function which takes an array with 10 elements as an input and adds 2 to all odd numbers and 4 to all even numbers present in the array.

```
Input Array - 5 2 4 3 6 1 7 8 9 10
Output Array - 7 6 8 5 10 3 9 12 11 14
```

Any Questions?

Thank You for attending!

Contact us regarding any questions through email

nandakishor2010608@ssn.edu.in
nitheesh2010343@ssn.edu.in