Sub: Chapter-10: User-defined Function

Day
Time: Date: / /

Write down three rules to pass an array in a function.

→ An array is a fixed-size sequenced collection of elements of the same data type.

There are three rules to pass an array to a function. And the following rules are:

1. The function must be called by passing only the name of the array.
2. In the function definition, the formal parameter must be an array type; the size of the array does not need to be specified.
3. The function prototype must show that the argument is an array.

What is recursion? Explain with example.

→ Recursion is a process in which a function calls itself directly or indirectly and the corresponding function is called as recursive function.

Sub :

Day

Time : Date : / /

What is parameters and arguments?

→ Parameters: A parameter is the variable listed inside the parentheses in the function definition.

Arguments: An argument is referred to the values that are passed within a function when the function is called.

Example:

```
#include<stdio.h>
int sum(int a, int b) // function definition and
{                     // int a and int b are
  int res = (a+b);    // parameters.
  return res; // returning the addition
}
int main()
{
  int num1 = 10, num2 = 20;
  int result;
  result = sum(num1, num2);
  printf("The summation is %d", result);
  return 0; // sum is called with num1 and
}           // num2 as arguments.
```

Sub :

Day

Time : Date : / /

How to declare a function?

→ All function of a C program, must be declared. A function declaration (also known as function prototype) consists of four parts. And they are-

1. Function type (return type)
2. Function name
3. Parameter list
4. Terminating semicolon

They are coded in the following format:

function-type functionname (parameter list);

For example,

int mul (int m, int n);

There are some rules to declare a function. Following these rules:

1. The parameter list must be separated by commas.
2. The parameter name do not need to be the same in the proto-type declaration and the function definition.
3. ~~The types of function must match the types of parameters.~~
4. Use of parameters names in the declaration is optional.
5. If the function has no formal parameters, the list is written as (void).

Sub: Day Time: Date: / /

6. The return type is optional, when the function returns int type data.
7. The retype must be void if no value is returned.
8. When the declare types do not match with the types in the function definition, compiler will produce an error.
3. The types must match the types of parameters in the function definition, in number and order.

Write down some characteristics of modular programming.

→ Modular programming is the process of subdividing a computer program into separate sub-programs. It is applied to the design and development of software systems. Some characteristics of modular programming are as follows:

1. Each module should do one thing.
2. Communication between modules is ~~ab~~allowed only by a calling module.
3. A module can be called by one and only one higher module.
4. No communication can take place directly between modules that do not have calling-called relationship.
5. All modules are designed as single-entry, single exit systems using control structures.

Sub :

Day

Time : Date : / /

4. List the categories of function with example.

→

Sub: _____ Day | Time: Date: / /

Q. Write a program to calculate the standard daviation of an array of values. The array elements are read from the terminal. Use function to calculate standard daviation and mean.

→ Standard daviation = $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\bar{x}-x_i)^2}$

where, $\bar{x}$ is mean of values.

Program:

```
#include<stdio.h>
#include<math.h>
#define SIZE 5
float std_dev(float a[], int);
float mean(float a[], int n);
int main()
{
 float value[SIZE];
 int i;
 printf("Enter %d float values\n", SIZE);
 for(i=0; i<SIZE; i++)
     scanf("%f", &value[i]);
 printf("std deviation is %f\n", std_dev(value, SIZE));
 return 0;
}
float std_dev(float a[], int n){
 int i;
 float x, sum = 0.0;
 x = mean(a, n)
 for(i=0; i<n; i++)
     sum += (x-a[i]) * (x-a[i]);
 return (sqrt(sum/(float)n));
}
```

Sub :

Day

Time : Date : / /

```
float mean (float a[], int n){
  int i;
  float sum = 0.0;
  for(i=0; i<n; i++)
     sum = sum + a[i];
  return (sum/(float)n);
}
```

Input :

Enter 5 float values

35.0 67.0 79.5 14.20 55.75

Output:

Std. deviation is 23.231582