**Assignment(1)**

**ClassPGDCA-I(Sem-I)**

**Subject: Computer Programming Using C**

**Topics:**Parameter Passing Techniques,C language and benefits of C language

**Submitted To:**

Rajveer Singh

Prof.of Comp.Sci.Department

**Ques1:What is C Language and the benefits of C language over other programming languages.**

**Ans:** The C language is a high-level, general-purpose programming language. It provides a straightforward, consistent, powerful interface for programming systems. That's why the C language is widely used for developing system software, application software, and embedded systems.

The C programming language has been highly influential, and many other languages have been derived from it. For example, C++ and Java are two popular modern dialects of C.

And C is an excellent choice for system programming, for example, developing operating systems, compilers, and network drivers. Despite its popularity, C is not without its criticisms. Some have argued that its syntax could be more complex and easier to learn, while others have noted its lack of standardization as a significant issue. Nevertheless, C remains a widely used and influential language and will probably continue for many years.

**Benefits of C language over other Programming Languages:**

C is a powerful programming language that offers several benefits over other languages.

* C is a universal language that can be used for various applications.
* C is a very efficient language that can write code that is both fast and reliable.
* C is a portable language, meaning that code written in C can be easily compiled and run on various platforms.
* C is a well-established language with a large and active community of developers constantly working on improving and creating new tools and libraries.

**Ques2:Describe the Methods Of Parameter Passing Techniques in C Language?**

There are different ways in which parameter data can be passed into and out of methods and functions. Let us assume that a function *B()* is called from another function *A()*. In this case *A* is called the ***“caller function”*** and *B* is called the ***“called function or callee function”***. Also, the arguments which *A* sends to *B* are called *actual arguments* and the parameters of *B* are called *formal arguments*.

**Terminology**

* **Formal Parameter :** A variable and its type as they appear in the prototype of the function or method.
* **Actual Parameter :** The variable or expression corresponding to a formal parameter that appears in the function or method call in the calling environment.
* **Modes:**
  + **IN:** Passes info from caller to callee.
  + **OUT:** Callee writes values in caller.
  + **IN/OUT:** Caller tells callee value of variable, which may be updated by callee.

**Important methods of Parameter Passing**

1. **Pass By Value:** This method uses *in-mode* semantics. Changes made to formal parameter do not get transmitted back to the caller. Any modifications to the formal parameter variable inside the called function or method affect only the separate storage location and will not be reflected in the actual parameter in the calling environment. This method is also called as***call by value***.

|  |
| --- |
| // C program to illustrate  // call by value  #include <stdio.h>    void func(int a, int b)  {      a += b;      printf("In func, a = %d b = %d\n", a, b);  }  int main(void)  {      int x = 5, y = 7;        // Passing parameters      func(x, y);      printf("In main, x = %d y = %d\n", x, y);      return 0;  } |

Output:

In func, a = 12 b = 7

In main, x = 5 y = 7

Languages like C, C++, Java support this type of parameter passing. [Java in fact is strictly call by value.](https://www.geeksforgeeks.org/g-fact-31-java-is-strictly-pass-by-value/)  
**Shortcomings:**

* + Inefficiency in storage allocation
  + For objects and arrays, the copy semantics are costly

1. **Pass by reference(aliasing):** This technique uses *in/out-mode* semantics. Changes made to formal parameter do get transmitted back to the caller through parameter passing. Any changes to the formal parameter are reflected in the actual parameter in the calling environment as formal parameter receives a reference (or pointer) to the actual data. This method is also called as **call by reference**. This method is efficient in both time and space

|  |
| --- |
| // C program to illustrate  // call by reference  #include <stdio.h>    void swapnum(int\* i, int\* j)  {      int temp = \*i;      \*i = \*j;      \*j = temp;  }    int main(void)  {      int a = 10, b = 20;        // passing parameters      swapnum(&a, &b);        printf("a is %d and b is %d\n", a, b);      return 0;  } |

Output:

a is 20 and b is 10