BCSE102L- Structured and object-oriented programming

Module 2

Dr. P.Keerthika
Associate Professor
School of Computer Science & Engineering
VIT, Vellore.

BCSE102L- Structured and object-oriented programming

Module:1	**C Programming Fundamentals**	**2 hours**
Variables - Reserved words – Data Types – Operators – Operator Precedence - Expressions - Type Conversions - I/O statements - Branching and Looping: if, if-else, nested if, if-else ladder, switch statement, goto statement - Loops: for, while and do…while – break and continue statements.		
Module:2	**Arrays and Functions**	**4 hours**
Arrays: One Dimensional array - Two-Dimensional Array – Strings and its operations. User Defined Functions: Declaration – Definition – call by value and call by reference - Types of Functions - Recursive functions - Storage Classes - Scope, Visibility and Lifetime of Variables.		
Module:3	**Pointers**	**4 hours**
Declaration and Access of Pointer Variables, Pointer arithmetic – Dynamic memory allocation – Pointers and arrays - Pointers and functions.		
Module:4	**Structure and Union**	**2 hours**
Declaration, Initialization, Access of Structure Variables - Arrays of Structure - Arrays within Structure - Structure within Structures - Structures and Functions – Pointers to Structure -		
Module:5	**Overview of Object-Oriented Programming**	**5 hours**
Features of OOP - Classes and Objects - "this" pointer - Constructors and Destructors - Static Data Members, Static Member Functions and Objects - Inline Functions – Call by reference - Functions with default Arguments - Functions with Objects as Arguments - Friend Functions and Friend Classes.		
Module:6	**Inheritance**	**5 hours**
Inheritance - Types of Inheritance: Single inheritance, Multiple Inheritance, Multi-level Inheritance, Hierarchical Inheritance - Multipath Inheritance - Inheritance and constructors.		
Module:7	**Polymorphism**	**4 hours**
Function Overloading - Operator Overloading – Dynamic Polymorphism - Virtual Functions - Pure virtual Functions - Abstract Classes.		
Module:8	**Generic Programming**	**4 hours**
Function templates and class templates, Standard Template Library.		
	Total Lecture hours:	**30 hours**

BCSE102L- Structured and object-oriented programming – Text Books and Reference Books

Text Book(s)	
1.	Herbert Schildt, C: The Complete Reference, 4th Edition, McGraw Hill Education, 2017
2.	Herbert Schildt, C++: The Complete Reference, 4th Edition, McGraw Hill Education, 2017.
Reference Books	
1.	Yashavant Kanetkar, Let Us C: 17th Edition, BPB Publicaitons, 2020.
2.	Stanley Lippman and Josee Lajoie, C++ Primer, 5th Edition, Addison-Wesley publishers, 2012.

BCSE102P- Structured and object-oriented programming Laboratory

Indicative Experiments	
1.	Programs using basic control structures, branching and looping
2.	Experiment the use of 1-D, 2-D arrays and strings and Functions
3.	Demonstrate the application of pointers
4.	Experiment structures and unions
5.	Programs on basic Object-Oriented Programming constructs.
6.	Demonstrate various categories of inheritance
7.	Program to apply kinds of polymorphism.
8.	Develop generic templates and Standard Template Libraries.

Text Book(s)	
1.	Robert C. Seacord, Effective C: An Introduction to Professional C Programming, 1st Edition, No Starch Press, 2020.
Reference Book(s)	
1.	Vardan Grigoryan and Shunguang Wu, Expert C++: Become a proficient programmer by learning coding best practices with C++17 and C++20's latest features, 1st Edition, Packt Publishing Limited, 2020.

BCSE102L- Structured and Object-Oriented Programming

- **Module-2: ARRAYS AND FUNCTIONS**
  - **Arrays – 1D & 2D**
  - **Strings and its operations**
  - **User defined Functions**
    - **Declaration and Definition**
    - **Call by Value and Call by reference**
    - **Types of Functions- Recursive Functions**
  - **Storage Classes**
    - **Scope, Visibility and lifetime of variables**

STORAGE CLASSES

- Variables in C differ in behaviour with respect to place when compared with other languages.
- **SCOPE OF THE VARIABLE:**
  - Range within which the variable will be active in a program is known as the scope of a variable.
  - If a variable is used outside the program then the scope of the variable is limited.
  - Scope of the variable determines over what region of the program a variable is actually available for use.
- Variables inside a function are classified into two types
  - **Local Variable-** value of the variable is valid within the function itself
  - **Global Variable-** Variables declared before main function is called Global variable. It can be used in any part of a program

Scope, Visibility & Lifetime

```
int a;
void main()
{
        char b ;
        {
                float c;
        }
}
```

Scope of a
Visibility of a
Lifetime of a

Scope of b
Visibility of b
Lifetime of b

Scope of c
Visibility of c
Lifetime of c

Categories of Storage Classes

- **Storage class of a variable tells the compiler**
  - Storage area of the variable
  - Initial value of the variable if not initialized
  - Scope of the variable
  - Life of the variable that is how long the variable would be active in the program.

- **Four Categories**
  - Automatic Variables
  - Static Variables
  - External Variables
  - Register Variables

Automatic Variables/ Local /Internal

- Variables declared inside a block and local to block in which declared are said to be **Automatic** variables. These variables can be accessed by block in which they are declared. Another block cannot access its value.
- Compiler treat variable declared inside block as automatic variable by default. Automatic variables are stored in memory. All variables declared inside the function is **auto** by default.

```
main()
{
        int number;
        ----------;
        ----------;
}
```

```
main()
{
        auto int number;
        ----------;
        ----------;
}
```

External Variables

- Variables that are active throughout the entire program are called as **external variables (global variables)**. External variables are declared outside the function body.
- Variable retain the value throughout the execution of a program. This storage class can be accessed by any function in same or different program file and change its value.

```
int number; float length=6.2;
main()
{
        -------;
}
function1()
{
        -------;
}
function2()
{
        ---------;
}
```

External Variables

```
int fun1(void);   // Other way of declaring nil arguments
int fun2(void);
int x;                              /*GLOBAL*/
main()
{
        x=10;                       /*GLOBAL */
        printf("x=%d\n", x);
        printf("x=%d\n", fun1());
        printf("x=%d\n", fun2());
}
Int fun1(void)
{
        x=x+10;    /*GLOBAL*/
}
int fun2(void)
{
        int x ;                     /*LOCAL*/
        x=1 ;
        return(x) ;
}
```

```
x=10
x=20
x=1
```

External Variables

```
main()
{
        y=5;
        --------;
        ---------;
}
int y;                  /*Global Declaration*/

func1()
{
        y=y+1;
}
```

```
main()
{
        extern int y;   /* External Declaration*/
        --------;
        ---------;
}
func1()
{
        extern int y;
        y=y+1;
}
int y;                  /*Global Declaration*/
```

Static Variables

- Static variable may be either an internal type or an external type depending on the place of declaration.
- Global and Local variable can be declared static. Static variables are initialized only once when they are compiled in a program.
- Variable can be declared using the keyword **static**.
- **Internal Static Variables** - They are local to the block in which declared. It exists until the end of the program. (**Note: Like an auto variables**). When the program is closed the function associated with that program is also excited and whenever it is visited again the same value exists.
- **External static variable** is declared outside of all functions. It is made available to all functions in a C program.

Static Variables

```
void start(void);
void main()
{
        int i;
        for(i=1;i<3;i++)
                start();

}
void stat(void)
{
        static int x=0; // Initialized once
        x=x+1;
        printf("x=%d\n",x);
}
```

Output??

Register Variables

- Register is a special storage area in a Central Processing Unit (CPU). There are 8 registers available inside a Computer. Register variable can be accessed only by block in which it is declared. It cannot be accessed by any other function.
- Register variable declared using keyword **register**. Both Local variable and formal parameter can be declared as a register. Register is used to increase the execution speed. Only integer or char variables are declared as register in most of the compilers but ANSI C supports all the data types.

Register Variables

```
void main()
{
        register int x;
        for(x=1;x<=10;x++)
        printf("%d",x);
        printf("%u",&x); // Not able to get the address
}

int main()
{       int i = 10;
        register int *a = &i;
        printf("The value of pointer : %d",*a);
        return 0;
}
```

VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Thank You