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**

STRING AND ITS OPERATIONS

- A string is a sequence of character that is treated as single data item.
- In C language the group of characters, digits and symbols enclosed within quotation marks are called as strings.
- Character strings are often used to build meaningful and readable programs. The common operations performed on character strings include
  - Reading and writing strings
  - Combining strings together
  - Copying one string to another
  - Comparing strings for equality
  - Extracting a portion of string

Declaration & Initialization of Strings

- C does not support strings as a data type. But it allows us to represent strings as character arrays.
- General Form

char string_name[size];

- **Examples:**

char city[10];

char name[20];

- When the Compiler assigns a character string to a character array, it automatically supplies a null character ('\0') at the end of the string. Therefore the size should be equal to the maximum number of characters in the string plus one.

char city [9] = "NEW YORK";

char city [9] ={'N','E','W',' ','Y','O','R','K','\0'};

Reading strings from the terminal

- Scanf can be used with **%s** format specification to read in a string of characters.

  scanf(%s", city); (Note: & not required)

  scanf(%ws", city);

- **Consider the Following statements:**

  char name[10];

  scanf ("%5s", name); // Hai // Welcome

H	A	I	\0	?	?	?	?	?	?

W	E	L	C	O	\0	?	?	?	?

What is the problem in %s and %ws??

Reading strings from the terminal

- Scanf with **%s or %ws** can read only strings without white spaces.
- Cannot be used for reading a text containing more than one word.
- C supports a format specification known as the **edit set conversion code % […]** that can be used to read a line containing a variety of characters, including white spaces.

```
char line[80];
scanf("%[^\n]",line];
printf("%s",line);
```

Using getchar and gets functions

- **Getchar**- read a single character from the terminal.

```
char ch;
ch=getchar();
```

- **Gets**- reading a string of text containing white spaces

```
gets(str);
```

- Reads character into **str** through keyboard until a new-line character is encountered and then appends a null character to the string. It does not skip white spaces.

For example:

```
char line [80];
gets(line);
printf("%s", line);
```

- Reads a line of text from the keyboard and displays it on the screen.

Using putchar and puts functions

- **Putchar**- function to output the values of character variables.

```
char ch='A';
putchar(ch); //  printf("%c",ch);
```

- Also we can use this function repeatedly to output a string of character stored in an array as like the following example.

```
char name[6]="HAI"
for (i=0;i<5;i++)
        putchar (name[i]);
```

- **Puts-**convenient way of printing string values

```
puts(str);
```

// **str** is a string variable containing a string value

STRING LIBRARY FUNCTIONS

Functions	Description
Strlen()	Determines the length of a string
Strcpy()	Copies a string from source to destination
Strncpy()	Copies character of a string to another string up to specified length
Stricmp()	Compares characters of two strings.
Strcmp()	Compares two strings
Strncmp()	Compares characters of two strings upto the specified length
Strnicmp()	Compares characters of two strings upto the specified length, Ignores case
Strlwr()	Converts uppercase characters of a string to lowercase
Strupr()	Converts lowercase character of a string to uppercase

Functions	Description
Strudp()	Duplicates a string
Strchr()	Determines the first occurrence of a given character in a string
Strrchr()	Determines the last occurrence of a given character in a string
Strstr()	Determines the first occurrence of a given string in another string
Strcat()	Appends source string to destination string
Strncat()	Appends source string to destination string upto the specified length
Strrev()	Reverses all character of a string
Strset()	Sets all character of a string with a given argument or symbol
Strnset()	Sets specified numbers of characters of a string with a given argument or symbol
Strspn()	Finds upto what length two strings are identical
Strpbrk()	Searches the first occurrence of the character in a given string and then displays the string starting from that character

Strlen() function

- **Strlen() function** counts the number of character in a given string.

strlen(string);

```
int main()
{
        char text[20];
        int len;
        printf("Enter the text:\n");
        gets(text);
        len=strlen(text);
        printf("length of the string=%d",len);
        return 0;
}
```

OUTPUT

```
Enter the text
Welcome
Length of the string: 7
```

Finding string length using Looping

```
int main()
{
        char text[20];
        int len=0, i=0;
        printf("Enter the text:\n");
        gets(text);
        while(text[i]!='\0')
        {
        len=len+1;
        i++;
        }
        printf("length of the string=%d",len);
        return 0;
}
```

OUTPUT

Enter the text
Welcome
Length of the string: 7

Strcpy() function

- **strcpy()** function almost like a string-assignment operator. This function copies the contents of one string to another

strcpy(string1,string2);

```
int main()
{
        char original[20],duplicate[20];
        printf("enter the string:");
        gets(original);
        strcpy(duplicate,original);
        printf("\n original string:%s",original);
        printf("\n  duplicate string:%s", duplicate);
}
```

OUTPUT

```
Enter the string:Welcome
Original String:Welcome
Duplicate String:Welcome
```

Finding string COPY using Looping

```
int main()
{
        char text1[20], text2[20];
        int  i;
        printf("Enter the string:\n");
        gets(text1);
        for(i=0;text1[i]!='\0';i++)
        {
        text2[i]=text1[i];
        }
        printf("Original string =%s \n",text1);
        printf("string copied =%s",text2);
        return 0;
}
```

OUTPUT

Enter the String: Welcome
Original String: Welcome
String Copied: Welcome

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

Thank You