Programming Definitions

1. Problem Solving:

Problem solving is a process of identifying a problem and finding the best solution for it.

2. Program:

A set of instructions that tells a computer what to do is called program.

Advantages of Computer Program:

1.A computer program can solve many problems by giving instructions to computer.

2.A program can process a large amount of data easily.

3.It can display the results in different styles

3. Algorithms & Pseudo Code:

An algorithm is a step by step procedure to solve a problem.

Properties:

1. The given problem should be broken down into simple and meaningful steps.
2. The steps should be numbered sequentially.
3. The steps should be descriptive and written in simple English.

Advantages:

1. Reduced complexity
2. Increased Flexibility
3. Ease of Understanding

4. Flowchart:

Flowchart is combination of two words flow and chart. A chart consists of different symbols to display information about any program. Flow indicates the direction of processing that takes place in the program.

5. Programming Language:

A set of words, symbols and codes used to write programs is called program language.

Types:

1. Low level language

2. High level language

6. Low Level Language:

These languages are near to computer hardware and far from human languages. Computer can understand these languages easily. Writing a program in low level languages requires a deep knowledge of the internal structure of computer hardware.

Types:

1. Machine Language

2. Assembly language

7. Machine Language:

A type of language in which instructions are written in binary form is called machine language. It is only language that is directly understood the computer. It is fundamental language of the computer.

8. Assembly Language:

It is the type of language in which symbols are used instead of binary code. These symbols are called mnemonics. For example **Sub** instruction is used to subtract two numbers.

9. High Level Language:

A type of language that is close to human languages is called high level language. It is easy to understand. Instructions of these languages are written in English like words such as **input** and **print** etc.

Types:

1. Procedural Language

2. Object Oriented Language

3. Procedural Language

10. Procedural Languages:

Procedural languages are also known **3GL** Language. In these languages program is predefined set of instructions. Computer executes these instructions in the same sequence in which they are written. Each instruction tells the computer what to do and how to do. Some procedural are **(FORTAN, BASIC, COBOL, PASCAL, C).**

11. Object Oriented Language:

OOP is a technique in which programs are written on the basis of objects. An object is a collection of data and functions. Object may represent a person, thing or place in real world. Some OOP languages are **(C++, Java).**

12. Non Procedural Languages:

These are also known as **4GL** Language. In this language user only need to tell the computer what to do not how to do. These languages accerlate program process and reduce coding errors. Some Non procedural languages are **(SQL and RPG).**

13. Source Code:

A program written in high level language is called source code. Computer cannot understand the statements of high level language. It converts these instructions in machine code then executed.

14. Object Code:

An object code is a program that is translated by a language processor. It is also called machine code. Computer can understand object code easily.

15. Language Processor:

Computer understands only machine language. A program written in high level or assembly language cannot be run on a computer directly. It must be converted into machine language before execution. Language processor is a software that convers these program into machine language.

Types of Language Processor:

1. Compiler **2.Interpreter** **3.Assembler**

16. Compiler:

A compiler is a program that converts the instruction of a high level language into machine language as a whole.

17. Interpreter:

Interpreter is a program that converts one statement of a program at one time.

18. Assembler:

An assembler is a translating program that translates the instruction of a assembly language into machine language.

19. Basic Structure of C Program:

The format of writing program in C is called its Structure. It consists of following parts.

1. Preprocessor directive

2. Main( ) function

3. Program body.

20. Preprocessor directive:

Preprocessor directive is an instruction given to the compiler before the execution of actual program. It modifies C source program before execution. It is also known as compiler directive.

21. Include Preprocessor:

Include preprocessor directive is used to include header files in the program. The syntax of using the directive is as follows:

#include<stdio.h>

The above statement tells the compiler to include the file **stdio.h** in source program before compiling it.

22. Header Files:

Header files are collection of standard library functions to perform different tasks. Many header files are used in a program. The extension of header file is **.h**. The "**include"** preprocessor directive is used to include header files in programs.

Syntax: #include<header_file_name>

Example: #include<stdio.h>

23. Main( ) Function:

The main() function is the place where the execution of a C program starts. When the program is executed, the controls entered the main function and starts executing its statements. Each program must contain main() function.

Syntax:

```
main()
{
 body of main function
}
```

24. Token:

Token is a language element that is used to form a statement. C statement may consist of different tokens. Different types of tokens are as follows:

25. Keyword:

Keyword is a word in C language that has a predefined meaning and purpose. The meaning and purpose of a keyword is defined by the developer of the language.

Example: for double if

26. Identifiers:

Identifier is the name of a variable or function etc. It is also called user defined word.

Example: Student_name , item20 etc.

27. Constants:

Constant is a quantity that cannot be changed during execution of a program.

Example: 213, 520 ,420.77

28. String Literals:

A collection of characters written in double quotations is called string or string literal.

Example: "Hasnain is CR of MIT-Class"

29. Operators:

Operator is a symbol that performs some operation. It acts on different operands.

Example: + - * \ %

30. Punctuators:

Punctuator is a symbol that is used to separate two tokens.

Example: [] {} () , :

31. White Spaces:

White spaces are used in a program to increase readability. Different types of white spaces are include space,tab and carriage return. C compiler ignores the white spaces.

32. Debugging:

An error in a computer program is known as bug. The process of finding and removing bugs is known as debugging.

33. Syntax Error:

A collection of rule to writing programs in a programming language is known as syntax. Syntax error is a type of error that occurs when an invalid statement is written in a program. Important causes of syntax error are as follows:

The statement terminator is missing at the end of statement.

A misspelled keyword.

Any of the delimiter is missing.

34. Logical Error:

A type of error that occurs due to poor logic of the programmer is called logical error. A statement with logical error may produce unexpected and wrong results in the program.

Example: Using wrong condition such as a<5 instead of a>5.

35. Run time Error:

A type of error that occurs during the execution of program is known as run-time error. It is caused when a statement directs the computer to execute an illegal operation such as dividing a number by zero.

36. Data type:

The data type defines a set of values and a set of operations on those values. The data and its type are defined before designing the actual program used to process the data. Three commonly used data types are **Int data type, float data type and char data type.**

37. Integer Data type:

Integer data is a numeric value with no decimal point or fraction. It includes both positive and negative values.

Example: 100,25,-60,-500 etc.

38. Int Data type:

Int data type is used to store integer values. It takes **two** or **four** bytes in memory depending on the computer and compiler being used. Its range is from **-32768 to 32767.**

39. Short int Data type:

Short int data type is used to store integer values. Its take **two** bytes in memory.Its range is from **-32768 to 32767.**

40. Unsigned int data type:

Unsigned int data type is used to store only positive integer values.It takes **two** bytes in memry.Its range is from b **0 to 65535.**

41. Long int Data type:

Long int data type is used to store large integer values. It takes four bytes in memory.Its range is from **-2,147,483,648 to 2,147,483,647.**

42. Unsigned long int data type:

Unsigned long int data type is used to store large positive integer values. It takes **four** bytes in memory. Its range is from **0 to 4,294,967,295.**

43. Real data type:

Real data is numeric value with decimal point or fraction. It is also called floating point number. It includes both positive and negative values.

Example: 10.5 ,110.3 , -12.5 etc.

44. Float data type:

Float data type is used to store real values. It takes **four** bytes in memory. Its range is from **3.4×10^{-38} to $3.4\times10^{+38}$**

45. Double data type:

Double data type is used to store large real values. It takes **eight** bytes in memory. Its range is from **1.7×10^{-308} to 1.7×10^{-308}**

46.Long double data type:

Long double data type is used to store very large real values. It takes **ten** bytes in memory. Its range is from **1.7×10^{-4932} to $1.7\times10^{+4932}$**

47. Character data type:

Char data type is used to store character value. It takes **1 byte** in memory. It is used to represent a letter, number or punctuations mark and a few other symbols.

48. Integer Overflow and Underflow:

Integer overflow occurs when the value assigned to an integer variable is more than maximum possible value. Integer Underflow occurs when the value assigned to an integer variable is less than possible minimum value.

Example: An integer variable can stored values from -32768 to 32767.If the assigned value is more than 32767, then it is **integer overflow.** If the assigned value is less than -32768 than is is **integer underflow.**

49. Precision: Precision is the number of digits after the decimal point.

50.Range: Range is the exponential power of 10.

Example of Precision & Range: In value 0.123456×10^{4} in this range is **4** and Precision is **6.**

51. Variable:

Variable is a named memory location or memory cell. It is used to store program's input data and its computational results during execution. The value of a variable may change during the execution of program. However, the name of variable cannot be changed.

52. Variable declaration:

The process of specifying the variable name and its type is called variable declaration.

Syntax: data_type variable_name**;**

Example: int a; float avg; char name;

53. Rules of variable declaration:

- ✓ Variable may include letters,number and underscore(_).
- ✓ The first character of variable must be letter or underscore.(9seconds,2kg are invalid)
- ✓ Blank spaces cannot used in variable name.
- ✓ Special symbols cannot be used as variable name.
- ✓ Reserved word cannot be used like **int,void,while etc.**
- ✓ A variable can be up to 31 characters.
- ✓ A variable can be declared only for one data type.

54. Variable Initialization:

The process of assigning a value to a variable at the time of declaration is known as variable initialization.

Syntax: data_type variable_name=value**;**

Example: int a=6; float avg=2.3; char grade='A';

55. Constants:

Constant is the quantity that cannot be changed during program execution. Following are two types of constants in C:

1. Literal Constants
2. Symbolic Constants

56. Literal Constant:

Literal constant is a constant that is typed directly in a program. It appears in the program whenever it is needed.

Types of Literal constants:

1. Integer constant: Integer constants are the numeric values without fraction or decimal point.

Example: 87 66 90 -55 -88 etc.

2. Floating point constant: Floating point constants are numeric values with decimal point or fraction. **Example 50.77F 10.55F**

3. Character constants: Any character written in single quotes is known as character constant.

Example: 'A' 'n' '+'

4. String constants: A set of characters written in double quotations is called string or string constants.

Example: "Pakistan" "456"

57. Symbolic Constants:

A symbolic constant is a name given to values that cannot be changed. After the constant is initialized, its value cannot be changed. A symbolic constant **PI** is used to indicate the value of PI which is 3.141593.

Types of Symbolic constants:

1. **Const Qualifier:** const qualifier is used to define a constant. The constant is declared by specifying its name and data type. The constant must be initialized with some value. The initialized value cannot be changed during program execution.
   Syntax: const data_type identifier=value;
   Example: const int N=100;
2. **define Directive:** define directive is also used to define a constant. The difference between const qualifier and define directive is that define directive does not specify the data type of the constant. Define directive starts with # symbol. It is not terminated with semicolon.
   Syntax: #define identifier value
   Example: #define PI 3.141593

58. Expression:

A statement that evaluates to a value is called an expression. An expression is consist of operators and operands.

Example: A+B (in this expression '+' is operator, A and B are variables used as operands in expression)

59. Operators:

Operators are the symbols that are used to perform certain operations on data. These include arithmetic operators, relational operators, logical operators, bitwise operators etc.

1. **Unary Operators:** A type of operator that works with one operand is known as unary operator.

Unary operators: -,++,--

Example: -a; N++; --X;

2. **Binary Operators:** A type of operator that works with two operands is known as binary operators.

Binary operators: +,-,%,/ etc.

Example: a+b; x/y;

60. Arithmetic Operators:

Arithmetic operator is a symbol that performs mathematical operation on data.

Arithmetic operators: +,-,*,/,%.

61. Assignment Operator:

A statement that assigns a value to a variable is known as assignment statement.

Syntax: Variable=expression;

Example: A=100; C=A+B;

62. Lvalue and Rvalue:

An **lvalue** is an operand that can be written on the left side of assignment operator =.It must always be a single value. An **rvalue** is an operand that can be written on the right side of assignment operator =.All lvalues can be used as rvalues but all rvalues cannot be used as lvalues. An expression X=5 is valid but 5=X is not valid.

63. Compound assignment Statement:

An assignment statement that assigns a value to many variables is known as compound assignment statement.

Example: A=B=100;

64. Compound assignment Operators:

Compound assignment operators are the combination of assignment operator with arithmetic operators.

Syntax: variable operator=expression;

Example: N+=10; (Is equal to N=N+10)

65. Increment Operator:

Increment operator is used to increase the value of a variable by **1**.It is denoted by symbol **++**.It is unary operator and works with single operand.

Prefix Form: In prefix form the increment operator is written before the variable as **++y;**

Postfix Form: In postfix form the increment operator is written after the variable as **y++;**

66. Decrement Operator:

Decrement operator is used to decrement the value of a variable by **1**.It is denoted by symbol **--**.It is unary operator and works with single operand.

Prefix Form: In prefix form the decrement operator is written before the variable as **--y;**

Postfix Form: In postfix form the decrement operator is written after the variable as **y--;**

Prefix Form	Postfix Form
A=++B OR A=--B In prefix form it increments the value of B by 1. Then it assigns the value of B to A	**A=B++ OR A=B--** In postfix form it firstly assign the value of B to A then increment or decrement the value of B by 1.

67. Type Casting:

The process of converting the data type of a value during execution is known type casting.Type casting can be performed in two ways:
Implicit type casting
Explicit type casting

68. Implicit Type Casting:

A type of type casting that is performed automatically by the C compiler is called type casting. The operands in arithmetic operation must be of similar types. If the data type operands are different, the operand with lower data type is converted into higher data type.

69. Explicit Type Casting:

Explicit casting is performed by programmer. It is performed by using cast operator. The cast operator tells the computer to convert the data type of a value.
Syntax: (type) expression;
Example: two float number x=10.5 and y=5.2 in explicit type casting it is written as int(x)% int(y) and convert the data type to integer.

70. Input and Output:

The process of giving something to compiler to computer is known as **input**. The input is mostly given by keyboard. The process of getting something from computer is known as **output**. The output is mostly display on monitor.

71. Escape Sequence:

Escape sequence are special characters used in control string to modify the format of output. These characters are not display in the output. These characters are combination with backslash "\".The backslash is called escape character. Different escape sequences used in C language are as follows:
\a This escape sequence is used to play beep during execution.
Example: printf("Lovely\aPakistan"); will display Lovely Pakistan
\b This escape sequence is used to insert the backspace in the output.
Example: : printf("Lovely\bPakistan"); will display LovelyPakistan
\f This escape sequence is used to insert a blank paper in the output.
Example: : printf("Lovely\fPakistan"); will display LovelyPakistan

\n This escape sequence is used to insert new line in output.

Example: : printf(“Lovely\nPakistan”); will display

Lovely

Pakistan

\r This escape sequence is used to move the cursor at the beginning of current line.

Example: : printf(“Lovely\rPakistan”); will display only Pakistan print in beginning of first line. Pakistan is printed that overwrites “Lovely”.

\t This escape sequence is used to insert a **Tab** in the output.

Example: printf(“Lovely\tPakistan”); will display Lovely Pakistan.

\’ This is used to display single quotes in the output like ‘Lovely Pakistan’

\” This is used to display double quotes in the output like “Lovely Pakistan”

72. Control Structure:

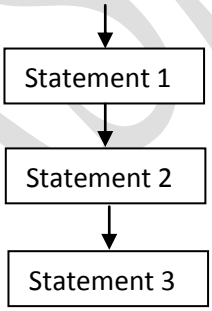
A statement used to control the flow of execution in a program is called control structure.

Types of control structure:

1. Sequence
2. Selection
3. Repetition
4. Function Call

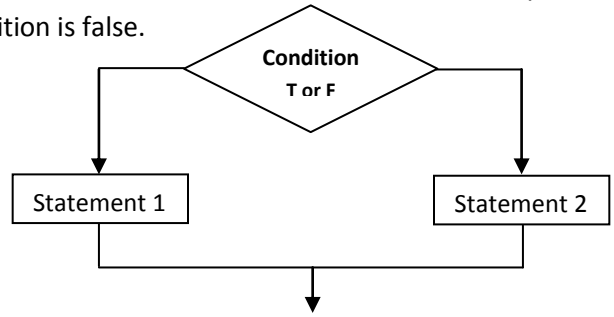
73. Sequence structure:

In sequence structure the statements are executed in the same order in which they are specified in program. The control flow one statement to other in a logical sequence. All statements are executed exactly once. It means no statement is skipped and no statement is executed more than once.



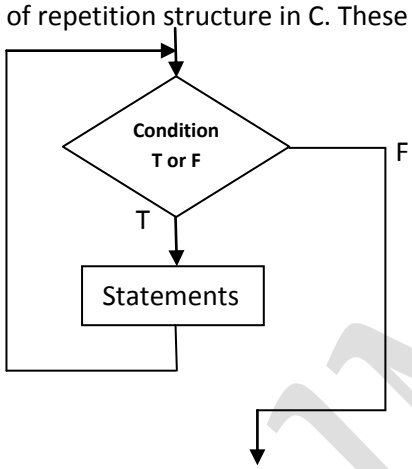
74. Selection Structure:

A selection structure selects a statement or set of statements to execute on the basis of a condition. In this structure statement or set of statements is executed when a particular condition is true and ignored when the condition is false.



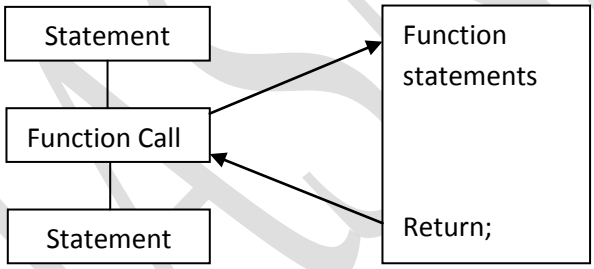
75. Repetition Structure:

A repetition structure executes a statement or set of statements repeatedly. It is also known as iteration structure. There are different types of repetition structure in C. These are while, do while and for loops.



76. Function Call:

Function call is a type of statement that moves the control to another block of code. The control returns back after executing all statements in the block. The remaining statements are executed immediately after the function call when the control is returned.



77. Relational Operators:

The relational operators are used to specify conditions in programs. A relational operator compares two values. It produces result as true or false. These are <,>,==,<=,>= and !=.

78. Relational Expression:

Relational expression is a statement that uses relational operators to compare two values. Some examples of relational operators are A<B, A>B, A>=B, A<=B, A==B and A!=B.

79. Compound Condition:

A type of comparison in which more than one conditions are evaluated is called compound condition. It is used to execute a statement or set of statements by testing many conditions. For example, a program inputs two numbers. It display OK if one number is greater than 100 and second number is less than 100.Compound condition is executed by using logical operators.

80. Conditional Operators:

Conditional operators is a decision making structure. It can be used in place of simple “if-else” structure. It is also called ternary operator because it uses three operands.

Syntax: (condition)! true-case statement; false-case statement;

81. If Statement:

If is a keyword in C language. If statement is a decision-making statement. It is the simplest form of selection constructors. It is used to execute or skip a statement or set of statements by checking a condition.

Syntax:

```
If(condition)
 Statement;
```

82. Limitation of If Statement:

If statement is the simplest selection structure but it is very limited in its use. The statement or set of statements is executed if the condition is true. But if the condition is false then nothing happens.

Example: A program should display ‘Pass’ if the student gets 40 or more marks and display fail if the student gets less than 40 marks. Simple if statement cannot be used to handle this situation.

83. if else statement:

If else statement is another type of if statement. It executes one block of statements when condition is true and the other when it is false. In any situation one block is executed and other is skipped.

Syntax:

```
If(condition)
{
Statement 1;
Statement 2;
.
.
.
.
Statement N;
}
else
{
Statement 1;
Statement 2;
.
Statement N;
}
```

85. Multiple if else if structure:

If else if statement can be used to choose one block of statements from many blocks of statements. It is used when there are many options ad only one block of statements should be executed on the basis of a condition.

Syntax:

```
If(condition)
{
block 1;
}
else If(condition)
{
block 2;
}
else If(condition)
{
block 3;
}
….
else
{
block N;
}
```

86. Nested if Structure:

An if statement within an if statement is called nested if statement. In nested structure, the control enters into the inner if only when the outer condition is true. Only one block of statements are executed and the remaining blocks are skipped automatically.

Syntax:

```
If(condition)
  If(condition)
{
Statements(s);
}
else
{
Statements(s);
}
else
{
statements(s);
}
```

87. Switch Structure:

Switch statement is another conditional structure. It is a good alternative of nested if-else. It can be used easily when there are many choices available and only one should be executed. Nested if becomes very difficult in such situation.

Syntax:

```
Switch(expression)
{
Case constant 1:
Statements(s);
break;
Case constant 2:
Statements(s);
break;
        .
        .
        .
Case constant N:
Statements(s);
break;
default:
statement(s);
}
```

88. Use of break in switch:

Break statement in each case label is used to exit from switch body. If break is not used,all case blocks coming after matching case will also be executed.

89. Use of default in switch:

The default block is used to execute a statement or set of statements when the result of expression in switch statement does not match with any case. It makes sure that switch executes properly if the selector is not in the range of case labels.

90. Loop:

A control structure that executes a statement or number of statements repeatedly is known as loop. Loops can be used to execute a statement or number of statements for a specified number of times.

91. While loop:

While loop is the simplest loop of C language. This loop executes one or more statements while the given condition remains true. It is useful when the number of iterations is not known in advance.

Syntax:

```
While(condition)
{
 Statement (s);
}
```

92. do-while loop:

This loop executes one or more statements while the given condition is true. In this loop, condition comes after the body of the loop. It is and important loop in a situation when the loop body must be executed at least once.

Syntax:

```
do
{
Statement 1;
Statement 2;
.
.
Statement N;
}
While(condition);
```

93. Difference between While and do-while loop:

While loop	do-while loop
In while loop condition comes before the body of the loop.	In do-while loop condition comes after the body of the loop.
If condition is false in the beginning while loop is never executed.	do-while loop is executed at least once even if condition is false in the beginning.

94. for loop:

For loop executes one or more statements for a specified number of times. This loop is also called counter controlled loop. It is most flexible loop. That is why it is the most frequently used loop by the programmers.

Syntax:

```
for(initialization;condition;increament\decrement)
{
Statement 1;
Statement 2;
.
.
Statement N;
}
```

95. Arrays:

A group of consecutive memory locations with same name and type is called an array.

96. Advantages\Uses of Arrays:

- Arrays can store a large number of values with single name.
- Arrays are used to process many values easily and quickly.
- The values stored in an array can be sorted easily.
- A search process can be applied on the arrays easily.

97. Declaring of an array:

The process of specifying array name,length, and its data type is called array declaration.

Syntax: data_type identifier[length];

Example: int marks[5];

98. Initialization of array:

The process of assigning values to array elements at the time of array declaration is called array initialization.

Syntax: data_type identifier[length]={List of values};

Example: int marks[5]={10,20,47,80,100};

99. Function:

A function is a named block of code that performs some action. The statements written in a function are executed when it is called by its name. Each function has a unique name. Functions are the building blocks of C programs.

100. Advantages of Function:

- Easier to code
- Easier to modify
- Easier to maintain
- Reusability
- Less programming time

101. User defined Function:

A type of function that is written by programmer is known as user defined function. A program may contain many user defined function. These functions are written according to the exact need of the user.

102. Built in Function:

A type of function that is available as a part of language is known as built in function or library function. These functions are ready made programs. These functions are stored in different header files built in function make programming faster and easier.

103. Function Declaration or Function prototype:

Function declaration is a model of a function. It is also called function prototype. It provides information to compiler about the structure of the function to be used in the program. It ends with a semicolon.

Syntax: Return_type function-name(parameters);

Example: int student(int,int);

104. Scope of a function:

The area in which a function can be accessed is known as the scope of a function. The scope of any function is determined by the place of function declaration.

105. Local Function:

A type of function that is declared within another function is called local function. A local function can be called within the function in which it is declared.

106. Global Function:

A type of function that is declared outside any function is called global function. A global function can be called from any part of the program.

107. Pass by value:

A parameter passing mechanism in which the value of actual parameter is copied to formal parameters of called function is known as pass by value.

108. Pass by Reference:

A parameters passing mechanism in which the address of the actual parameters is passed to the called function is known as pass by reference. The formal parameters is not created separately in the memory.

109. Pointers:

A pointer is a variable that is used to store a memory address. The reference operator is used to access the memory address of a variable and store it in a pointer.

110. Pointer declaration:

The method of declaring a pointer is same as declaring a simple variable. An asterisk '*' is used in the declaration that indicates that the variable is a pointer variable.

Syntax: data_type *var;

Example: int *p;

111. Pointer Initialization:

The process of assigning a memory address to a pointer at the time of declaration is called pointer initialization.

Syntax: data_type *p=&variable;

Example: int *p=&n;

112. String:

A collection of characters written in double quotations is called string or string constant.

Examples: "Pakistan" "123" "Islamabad"

113. String declaration:

C stores a string as an array of characters. An array is a group of consecutive memory location that can store same type of data.

Syntax: char array-name[length];

Example: char book[20];

114. String Initialization:

The process of assigning a value to a string at the time of their declaration is called string initialization.

Syntax: char array-name[length]=value;

Example: char book[20]="I love Pakistan";

115. Structure:

A structure is a collection of multiple data types that can be referenced with single name. It may contain similar or different data type. The data items in a structure are called structure elements, members or fields. The structures are used to join simple variables together to form complex variable.

116. Declaring a Structure:

A structure is declared b using the keyword **struct** followed by the structure name. The structure members are defined with their type inside the opening and closing braces { }. The closing braces ending with semicolon.

Syntax:

```
Struct struct_name
{
  Data_type identifier 1;
  Data_type identifier 2;
          .
          .
          .
 };
```

117. Defining structure variable:

The structure variable can be defined after the declaration of a structure. The process of defining a structure variable is same as defining a variable of basic types such as int and char.The definition tells the compiler to allocate a memory space for the variable. The compiler automatically allocates sufficient memory according to the elements of the structure.

Syntax: Struct_name identifier:

Example: Student Hasnain;

118. Accessing Members of structure variable:

Any member of a structure variable can be accessed by using dot operator. The name of structure variable is written on the left side of the dot. The name of member variable is written on right side of the dot.

Syntax: struct_var.Mem_var;

119. Initializing structure variables:

The process of assigning values to member variables of structure is called initialization of structure variable.

Syntax: struct_name identifier = {value1,value2,…….};

120. Recursion:

A programming technique in which a function calls itself is known as recursion. A function that calls itself is called recursion function.

121. File handling:

File Handling concept in **C** language is used for store a data permanently in computer. Using this concept we can store our data in Secondary memory (Hard disk). All **files** related function is available in stdio.h header **file**.

122. Files:

A file is a collection of related records. A record contains data about an entity. A file can be used to save any type of data.

123. Advantages of Files:

- Files can be store large amount of data permanently.
- Files can be updated easily.
- One file can be used by many programs for same input.
- Files save time and effort to input data via keyboard.

124. Text file:

A type of file that stores data as readable and printable characters is called text file. A source program of a C program is an example of text file.

125. Binary file:

A type of file that stores data as non-readable binary code is called binary file. An object file of C program is an example of binary file.

126. File Access Method:

The way in which a file can be accessed is called file access method. It depends on the manner in which data is stored in the files.

127. Sequential Access Method:

Sequential access method is used to access data in the same sequence in which it is stored in a file. This method reads and writes data in a sequence. In order to read the last record, it has to read all records stored before the last one.

128. Random Access Method:

Random access method is used to access any data item directly without accessing the preceding data. It is very fast access method as compared to sequential access method. A searching operation with random access method takes far less time to find a data item.

129. Declaring a file pointer:

In order to declare a file, we use a file pointer. A file pointer is a pointer variable that specifies the next byte to be read or written too. Every time a file is opened, the file pointer points to the beginning of the file. A file is declared as follows:

Syntax: FILE *pointer_name;

Example: FILE *p;

130. Opening and Creating a File

C language provides a number of functions to perform file handling. fopen() is used to create a new file or open an existing file. The syntax is as follows:

Syntax: p=fopen(“File_name”,”mode”); where ‘p’ is file pointer name.

Example: p=fopen(“student.txt”,”w”);

131. Closing a file:

A file needs to be closed after a read or write operation to release the memory allocated by the program. In C, a file is closed using the *fclose()* function. This returns 0 on success and EOF in the case of a failure.

Syntax: fclose(file-pointer_name);

Example: fclose(p);

Prepared By:

Hasnain Shoukat (Cr-MIT)