PIC Viva Notes

C basic commands:

C Basic commands	Explanation
#include <stdio.h>	This command includes standard input output header file(stdio.h) from the C library before compiling a C program
int main()	It is the main function from where C program execution begins.
{	Indicates the beginning of the main function.
/*_some_comments_*/	Whatever written inside this command “/* */” inside a C program, it will not be considered for compilation and execution.
printf(“Hello_World! “);	This command prints the output on the screen.
getch();	This command is used for any character input from keyboard.
return 0;	This command is used to terminate a C program (main function) and it returns 0.
}	It is used to indicate the end of the main function.

Pre-processor directive:

#include is a pre-processor directive in ‘C.’

#include <stdio.h> is the library where the function **printf** is defined**.** Before using this function, we have to first include the required file, also known as a header file (.h).

You can also create your own functions, group them in header files and declare them at the top of the program to use them. To include a file in a program, use pre-processor directive

```
#include <file-name>.h
```

Pre-processor directives are always placed at the beginning of the program.

Main Function:

In int main(), The empty parentheses indicate that this function does not take any argument, value or a parameter. The keyword void inside the parentheses does the same job. The keyword void means the function does not return any value, in this case, the last statement is always getch (). The keyword int means the function will return an integer value. In this case, the last statement should always return 0.

Comments:

A **comment** is an explanation or description of the source code of the program. It helps a developer explain logic of the code and improves program readability. At run-time, a comment is ignored by the compiler.

There are two types of comments in C:

1) A comment that starts with a slash asterisk /* and finishes with an asterisk slash */ and you can place it anywhere in your code, on the same line or several lines.

2) Single-line Comments which uses a double slash // dedicated to comment single lines

What is Token in C?

TOKEN is the smallest unit in a 'C' program. It is each and every word and punctuation that you come across in your C program. The compiler breaks a program into the smallest possible units (Tokens) and proceeds to the various stages of the compilation. C Token is divided into six different types, viz, Keywords, Operators, Strings, Constants, Special Characters, and Identifiers.

Keywords and Identifiers

In 'C' every word can be either a keyword or an identifier.

Keywords have fixed meanings, and the meaning cannot be changed. They act as a building block of a 'C' program. There are a total of 32 keywords in 'C'. Keywords are written in lowercase letters.

Keywords in C Programming Language			
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	short	float	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

An identifier is nothing but a user-defined name assigned to an element in a program. Each identifier must have a unique name. Following rules must be followed for identifiers:

1. The first character must always be an alphabet or an underscore.
2. It should be formed using only letters, numbers, or underscore.
3. A keyword cannot be used as an identifier.
4. It should not contain any whitespace character.
5. The name must be meaningful.

What is a Variable?

A variable is an identifier which is used to store some value. Constants can never change at the time of execution. Variables can change during the execution of a program and update the value stored inside it.

Following are the rules that must be followed while creating a variable:

1. A variable name should consist of only characters, digits and an underscore.
2. A variable name should not begin with a number.
3. A variable name should not consist of whitespace.
4. A variable name should not consist of a keyword.
5. 'C' is a case sensitive language that means a variable named 'age' and 'AGE' are different.

Data types

Following are the three data types:

1. Primitive data types
2. Derived data types
3. User-defined data types

There are five primary fundamental data types,

1. int for integer data
2. char for character data
3. float for floating point numbers
4. double for double precision floating point numbers
5. void

Array, functions, pointers, structures are derived data types.

Each data type differs from one another in size and range.

Data type	Size in bytes	Range
Char or signed char	1	-128 to 127
Unsigned char	1	0 to 255
int or signed int	2	-32768 to 32767
Unsigned int	2	0 to 65535
Short int or Unsigned short int	2	0 to 255
Signed short int	2	-128 to 127
Long int or Signed long int	4	-2147483648 to 2147483647
Unsigned long int	4	0 to 4294967295
float	4	3.4E-38 to 3.4E+38
double	8	1.7E-308 to 1.7E+308
Long double	10	3.4E-4932 to 1.1E+4932

Note: In C, there is no Boolean data type.

Format specifiers:

The format specifiers are used in C for input and output purposes. Using this concept the compiler can understand that what type of data is in a variable during taking input using the scanf() function and printing using printf() function. For function float (%f) and char (%c) and int (%d).

Constants:

Constants are the fixed values that never change during the execution of a program. They cannot be modified once declared. Use keyword ‘const’ or use #define as a pre-processor directive.

C Arithmetic Operators

An arithmetic operator performs mathematical operations such as addition, subtraction, multiplication, division etc on numerical values (constants and variables).

C Increment and Decrement Operators

C programming has two operators increment `++` and decrement `--` to change the value of an operand (constant or variable) by 1.

These two operators are unary operators, meaning they only operate on a single operand.

C Assignment Operators

An assignment operator is used for assigning a value to a variable. The most common assignment operator is =, followed by +=, -=, *=, /= , %=

C Relational Operators

A relational operator checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false, it returns value 0. Eg: <, >, ==, <=, >=, !=.

C Logical Operators

An expression containing logical operator returns either 0 or 1 depending upon whether expression results true or false. Logical operators are commonly used in decision making in C programming.

Operator	Meaning	Example
&&	Logical AND. True only if all operands are true	If c = 5 and d = 2 then, expression `((c==5) && (d>5))` equals to 0.

Operator	Meaning	Example
||	Logical OR. True only if either one operand is true	If c = 5 and d = 2 then, expression `((c==5) || (d>5))` equals to 1.
!	Logical NOT. True only if the operand is 0	If c = 5 then, expression `!(c==5)` equals to 0.

C Bitwise Operators

During computation, mathematical operations like: addition, subtraction, multiplication, division, etc are converted to bit-level which makes processing faster and saves power.

Bitwise operators are used in C programming to perform bit-level operations.

&, |, !, ^, ~, <<, >>.

* ^ Bitwise XOR
~ Bitwise complement (Transforms bit 0 to 1 and 1 to 0)
* Two's complement of a binary number is generated by adding one to the 1s complement of a binary number.

What is a Conditional Statement in C?

They are used to make decisions based on the conditions. They execute sequentially when there is no condition around the statements. If you put some condition for a block of statements, the execution flow may change based on the result evaluated by the condition.

1. If statement

2. If-else statement

This process is called decision making in 'C.' It is also called as branching as a program decides which statement to execute based on the result of the evaluated condition.

If statement

If statement is always used with a condition. The condition is evaluated first before executing any statement inside the body of If.

The condition evaluates to either true or false. True is always a non-zero value, and false is a value that contains zero.

if the value of test-expression is true, then the true block of statements will be executed.

If the value of test-expression if false, then the false block of statements will be executed.

In any case, after the execution, the control will be automatically transferred to the statements appearing outside the block of If.

Nested If-else Statements

When a series of decision (multipath decisions) is required, nested if-else is used. Nesting means using one if-else construct within another one.

Ternary operator

It is used to **execute** code based on the **result** of a **binary condition**. It takes in a binary condition as input, which makes it similar to an 'if-else' control flow block.

The Conditional Operator '?:' takes three operands to work, hence they are also called **ternary operators**.

```
variable = Expression1 ? Expression2 : Expression3
```

What is Loop in C?

Looping Statements in C execute the sequence of statements many times until the stated condition becomes false.

A loop in C consists of two parts a <u>body of a loop</u> and a <u>control statement</u>.

The control statement is a combination of some conditions that direct the body of the loop to execute until the specified condition becomes false. The purpose of the C loop is to repeat the same code a number of times.

Types of Loops in C

Depending upon the position of a control statement in a program, looping statement in C is classified into two types:

1. Entry controlled loop

2. Exit controlled loop

In an <u>entry control loop in C</u>**,** a condition is checked <u>before</u> executing the body of a loop. It is also called as a **pre-checking loop.**

In an <u>exit controlled loop</u>, a condition is checked <u>after</u> executing the body of a loop. It is also called as a **post-checking loop.**

The control conditions must be well defined and specified otherwise the loop will execute an infinite number of times.
The loop that does not stop executing and processes the statements number of times is called as an **infinite loop**. An infinite loop is also called as an "**Endless loop**."

Following are some characteristics of an infinite loop:

1. No termination condition is specified.
2. The specified conditions never meet.

‘C’ programming language provides us with three types of loop constructs:

Sr. No.	Loop Type	Description
1.	While Loop	In while loop, a condition is evaluated before processing a body of the loop. If a condition is true then and only then the body of a loop is executed.
2.	Do-While Loop	In a do…while loop, the condition is always executed after the body of a loop. It is also called an exit-controlled loop. Executes the body of the loop at least once even if the condition is false.
3.	For Loop	In a for loop, the initial value is performed only once, then the condition tests and compares the counter to a fixed value after each iteration, stopping the for loop when false is returned.

Syntax of For Loop in C:

- Initialization- The initial value of the for loop is performed only once.
- Checking Condition- The condition is a Boolean expression that tests and compares the counter to a fixed value after each iteration, stopping the for loop when false is returned.
- Incrementation/Decrementation- increases (or decreases) the counter by a set value.

Break Statement in C

The break statement is used mainly in the switch statement. It is also useful for immediately stopping a loop.

Continue Statement in C

When you want to skip to the next iteration but remain in the loop, you should use the continue statement.

What is Switch Statement in C?

Switch statement in C is a special type of if-else statement that tests the value of a variable and compares it with multiple cases. Once the case match is found, a block of statements associated with that particular case is executed.

If a case match is NOT found, then the default statement is executed, and the control goes out of the switch block.

We need to introduce a break statement in each case at the end of a switch statement. Or else, after it finds a case match, it will execute all cases until it reaches a break; statement.

Nested Switch in C

An inner switch is embedded in an outer switch. Also, the case constants of the inner and outer switch may have common values and without any conflicts.

Why do we need a Switch case?

There is one potential problem with the if-else statement which is the complexity of the program increases whenever the number of alternative path increases. If you use multiple if-else constructs in the program, a program might become difficult to read and comprehend. – confusing for even the developer.

The solution to this problem is the switch statement.

String in C is nothing but a collection of characters in a linear sequence. A single character is defined using single quote representation. A string is represented using double quote marks.

A C String is a simple array with char as a data type. to display a String in C, you need to make use of a character array. **The C compiler automatically adds a NULL character '\0' to the character array created.**

When we use scanf() to read, we use the "%s" format specifier without using the "&" to access the variable address because an array name acts as a pointer. As soon as scanf encounters a whitespace, it stops reading the string.

In order to read a string contains spaces, we use the gets() or fgets() function. It stops reading when a newline is reached (the Enter key is pressed) and ignoreds whitespaces.

```
fgets(name, 10, stdin);
```

The fgets() arguments are :

- the string name,
- the number of characters to read,
- stdin means to read from the standard input which is the keyboard.

The string library

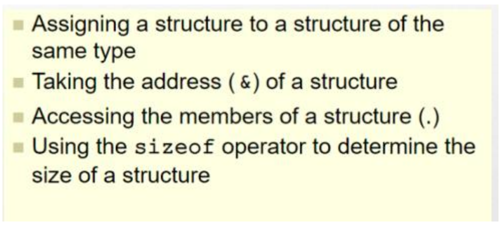
The standard 'C' library provides various functions (string handlers) to manipulate the strings within a program. All these handlers are present inside <string.h> header file.

Function	Purpose
strlen()	This function is used for finding a length of a string. It returns how many characters are present in a string excluding the NULL character.
strcat(str1, str2)	This function is used for combining two strings together to form a single string. It Appends or concatenates str2 to the end of str1 and returns a pointer to str1.
strcmp(str1, str2)	This function is used to compare two strings with each other. It returns 0 if str1 is equal to str2, less than 0 if str1 < str2, and greater than 0 if str1 > str2.

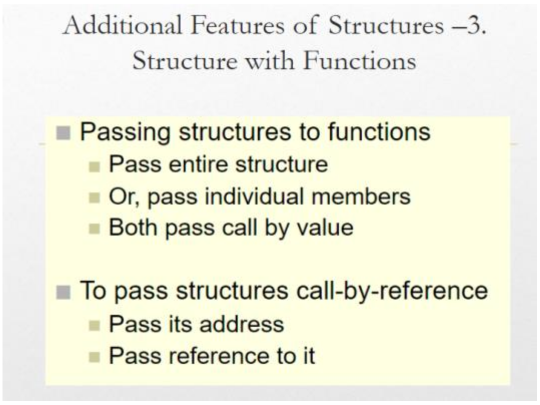
What is Structure?

Structure is a user-defined data type in C programming language that combines logically related data items of different data types together. All the structure elements are stored at contiguous memory locations.

Valid operations on Structures

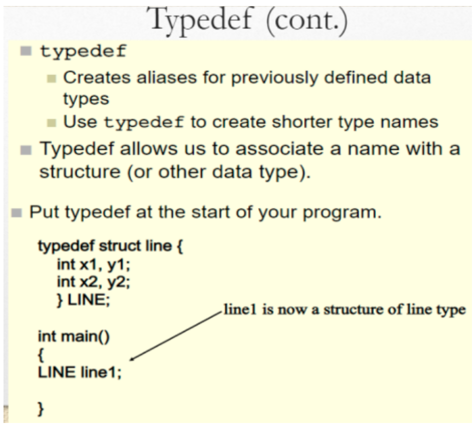


- Assigning a structure to a structure of the same type
- Taking the address (`&`) of a structure
- Accessing the members of a structure (.)
- Using the `sizeof` operator to determine the size of a structure



Additional Features of Structures –3. Structure with Functions

- Passing structures to functions
  - Pass entire structure
  - Or, pass individual members
  - Both pass call by value
- To pass structures call-by-reference
  - Pass its address
  - Pass reference to it



Typedef (cont.)

- `typedef`
  - Creates aliases for previously defined data types
  - Use `typedef` to create shorter type names
- Typedef allows us to associate a name with a structure (or other data type).
- Put typedef at the start of your program.

```
typedef struct line {
   int x1, y1;
   int x2, y2;
   } LINE;

int main()
{
LINE line1;

}
```

line1 is now a structure of line type

What is Union

Union is a user-defined data type, just like a structure. Union combines objects of different types and sizes together. The union variable allocates the memory space equal to the space to hold the largest variable of union. It allows varying types of objects to share the same location.

Example of Union in C Programming

```
#include <stdio.h>

union item
{
    int x;
    float y;
    char ch;
};

int main( )
{
    union item it;
    it.x = 12;
    it.y = 20.2;
    it.ch = 'a';

    printf("%d\n", it.x);
    printf("%f\n", it.y);
    printf("%c\n", it.ch);

    return 0;
}
```

Output:

```
1101109601
20.199892
a
```

In the above program, you can see that the values of x and y gets corrupted. Only variable ch prints the expected result. It is because, in union, the memory location is shared among all member data types.

Therefore, the only data member whose value is currently stored, will occupy memory space. The value of the variable ch was stored at last, so the value of the rest of the variables is lost.

Difference bw Structure and Union

Structure	Union
You can use a struct keyword to define a structure.	You can use a union keyword to define a union.
Every member within structure is assigned a unique memory location.	In union, a memory location is shared by all the data members.
Changing the value of one data member will not affect other data members in structure.	Changing the value of one data member will change the value of other data members in union.
It enables you to initialize several members at once.	It enables you to initialize only the first member of union.
The total size of the structure is the sum of the size of every data member.	The total size of the union is the size of the largest data member.
It is mainly used for storing various data types.	It is mainly used for storing one of the many data types that are available.
It occupies space for each and every member written in inner parameters.	It occupies space for a member having the highest size written in inner parameters.
You can retrieve any member at a time.	You can access one member at a time in the union.
It supports flexible array.	It does not support a flexible array.

Advantages of structure

- Structures gather more than one piece of data about the same subject together in the same place.
- It is helpful when you want to gather the data of similar data types and parameters like first name, last name, etc.
- It is very easy to maintain as we can represent the whole record by using a single name.
- In structure, we can pass complete set of records to any function using a single parameter.
- You can use an array of structure to store more records with similar types.

Advantages of union

- It occupies less memory compared to structure.
- When you use union, only the last variable can be directly accessed.
- Union is used when you have to use the same memory location for two or more data members.
- It enables you to hold data of only one data member.
- Its allocated space is equal to maximum size of the data member.

Disadvantages of structure

- If the complexity of IT project goes beyond the limit, it becomes hard to manage.
- Change of one data structure in a code necessitates changes at many other places. Therefore, the changes become hard to track.

- Structure is slower because it requires storage space for all the data.
- You can retrieve any member at a time in structure whereas you can access one member at a time in the union.
- Structure occupies space for each and every member written in inner parameters while union occupies space for a member having the highest size written in inner parameters.
- Structure supports flexible array. Union does not support a flexible array.

Disadvantages of union

- You can use only one union member at a time.
- All the union variables cannot be initialized or used with varying values at a time.
- Union assigns one common storage space for all its members.

C File management – Refer ppt for file handling also

A File can be used to store a large volume of persistent data. Following are the file management functions;

1. Creation of a file
2. Opening a file
3. Reading a file
4. Writing to a file
5. Closing a file

function	**purpose**
fopen ()	Creating a file or opening an existing file
fclose ()	Closing a file
fprintf ()	Writing a block of data to a file
fscanf ()	Reading a block data from a file
getc ()	Reads a single character from a file
putc ()	Writes a single character to a file
getw ()	Reads an integer from a file
putw ()	Writing an integer to a file
fseek ()	Sets the position of a file pointer to a specified location
ftell ()	Returns the current position of a file pointer
rewind ()	Sets the file pointer at the beginning of a file

```
fp = fopen ("file_name", "mode");
```

File Mode	**Description**
r	Open a file for reading. If a file is in reading mode, then no data is deleted if a file is already present on a system.

File Mode	Description
w	Open a file for writing. If a file is in writing mode, then a new file is created if a file doesn't exist at all. If a file is already present on a system, then all the data inside the file is truncated, and it is opened for writing purposes.
a	Open a file in append mode. If a file is in append mode, then the file is opened. The content within the file doesn't change.
r+	open for reading and writing from beginning
w+	open for reading and writing, overwriting a file
a+	open for reading and writing, appending to file

File is created in the same folder where you have saved your code.

```
FILE *fp;
fp  = fopen ("data.txt", "r");
fclose (fp);
```

fp is a file pointer which points to the type file.

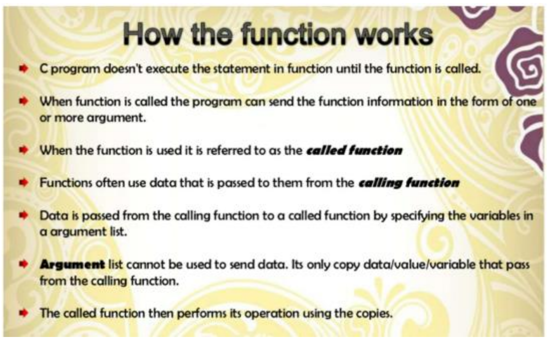
What is a Function in C?

It is a reusable block of code that makes a program easier to understand, test and can be easily modified without changing the calling program. Functions divide the code and modularize the program for better and effective results.

2 types: User Defined and Library function

C programming functions are divided into three activities such as,

1. Function declaration
2. Function definition
3. Function call



How the function works

- C program doesn't execute the statement in function until the function is called.
- When function is called the program can send the function information in the form of one or more argument.
- When the function is used it is referred to as the ***called function***
- Functions often use data that is passed to them from the ***calling function***
- Data is passed from the calling function to a called function by specifying the variables in a argument list.
- **Argument** list cannot be used to send data. Its only copy data/value/variable that pass from the calling function.
- The called function then performs its operation using the copies.

Function Arguments

A function's arguments are used to receive the necessary values by the function call. They are matched by position; the first argument is passed to the first parameter, the second to the second parameter and so on.

Variables which are declared inside a function are local to that block of code and cannot be referred to outside the function. However, variables which are declared outside all functions are global and accessible from the entire program.

Recursive Function

It is a function which calls itself and includes an exit condition in order to finish the recursive calls. Recursion works by "stacking" calls until the exiting condition is true. Used in Factorial of a number program, exit cond; fact=1;

What is Pointer in C?

It is a variable that stores address of another variable.
A pointer can be incremented/decremented, i.e., to point to the next/ previous memory location. The purpose of pointer is to save memory space and achieve faster execution time.
Int *y = &v;

Operator	Meaning
*	Serves 2 purpose 1. Declaration of a pointer 2. Returns the value of the referenced variable
&	Serves only 1 purpose • Returns the address of a variable

Advantages of Pointers in C

- Pointers are useful for accessing memory locations.
- Pointers provide an efficient way for accessing the elements of an array structure.
- Pointers are used for dynamic memory allocation as well as deallocation.
- Pointers are used to form complex data structures such as linked list, graph, tree, etc.

Disadvantages of Pointers in C

- Pointers are a little complex to understand.
- Pointers can lead to various errors such as segmentation faults or can access a memory location which is not required at all.
- If an incorrect value is provided to a pointer, it may cause memory corruption.
- Pointers are also responsible for memory leakage.
- Pointers are comparatively slower than that of the variables.

- Difficult to work with

Dynamic Memory Allocation in C

is manual allocation and freeing of memory according to your programming needs during runtime. Dynamic memory is managed and served with pointers that point to the newly allocated memory space in an area which we call the heap.
Now you can create and destroy an array of elements dynamically at runtime. The automatic memory management uses the stack, and the C Dynamic Memory Allocation uses the heap.

The <stdlib.h> library has functions responsible for Dynamic Memory Management.

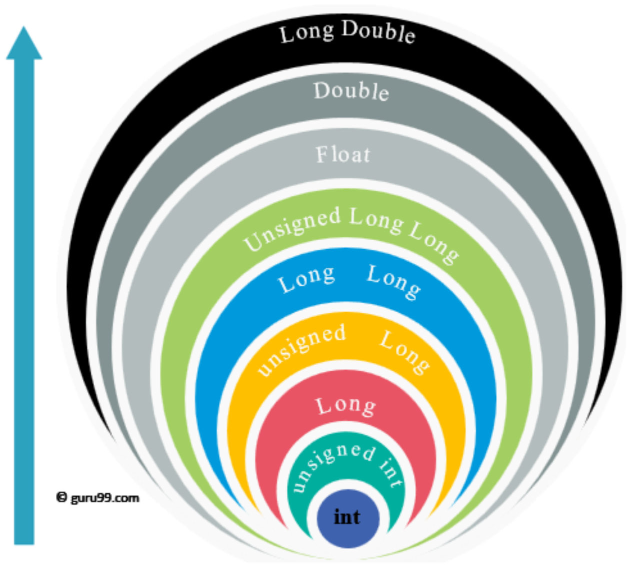
Function	Purpose
malloc()	Allocates the memory of requested size and returns the pointer to the first byte of allocated space.
calloc()	Allocates the space for elements of an array. Initializes the elements to zero and returns a pointer to the memory.
realloc()	It is used to modify the size of previously allocated memory space.
Free()	Frees or empties the previously allocated memory space.

```
ptr = (cast_type *) malloc (byte_size);
```

malloc()	**calloc()**
malloc() function creates a single block of memory of a specific size.	calloc() function assigns multiple blocks of memory to a single variable.
The number of arguments in malloc() is 1.	The number of arguments in calloc() is 2.
malloc() is faster.	calloc() is slower.
malloc() has high time efficiency.	calloc() has low time efficiency.
The memory block allocated by malloc() has a garbage value.	The memory block allocated by calloc() is initialized by zero.
malloc() indicates memory allocation.	calloc() indicates contiguous allocation.

Hierarchy for typecasting:

Type casting refers **to changing an variable of one data type into another**.


Long Double
Double
Float
Unsigned Long Long
Long Long
unsigned Long
Long
unsigned int
int
© guru99.com