Name :mahmoud hassan amin ibrahem Track : IOT Intake : 43

- What is the difference between Implicit and Explicit casting

Implicit	Explicit
-Implicit type casting means conversion of data types without losing its original meaning. This type of typecasting is essential when you want to change data types without changing the significance of the values stored inside the variable. -Implicit type conversion in C happens automatically when a value is copied to its compatible data type. During conversion, strict rules for type conversion are applied. If the operands are of two different data types, then an operand having lower data type is automatically converted into a higher data type.	-In implicit type conversion, the data type is converted automatically. There are some scenarios in which we may have to force type conversion. Suppose we have a variable div that stores the division of two operands which are declared as an int data type. -To force the type conversion in a situations, we use explicit type casting.

EX :
`#include<stdio.h>`
`int main(){`
`  short a=10; //initializing variable of short data type`
`  int b; //declaring int variable`
`  b=a; //implicit type casting`
`  printf("%d\n",a);`
`  printf("%d\n",b);`
`}`
	EX:
	`#include<stdio.h>`
	`int main()`
	`{`
	`  float a = 1.2;`
	`  //int b  = a; //Compiler will throw an error for this`
	`  int b = (int)a + 1;`
	`  printf("Value of a is %f\n", a);`
	`  printf("Value of b is %d\n",b);`
	`  return 0;`
	`}`

Summary

1. Typecasting is also called as type conversion
2. It means converting one data type into another.
3. Converting smaller data type into a larger one is also called as type promotion.
4. There are two type of type conversion: implicit and explicit type conversion in C.
5. Implicit type conversion operates automatically when the compatible data type is found.
6. Explicit type conversion requires a type casting operator.

- What is the difference between Pointer to Array and Array of Pointers:

Parameters	Pointer to an Array	Array of Pointers
Uses and Purposes	A user creates a pointer for storing the address of any given array.	A user creates an array of pointers that basically acts as an array of multiple pointer variables.
Alternative Names	It is alternatively known as an array pointer.	These are alternatively known as pointer arrays.
Allocation	One can allocate these during the run time.	One can allocate these during the compile time.
Initialization at Definition	You cannot initialize a pointer to the definition.	You can easily initialize an array at the definition level.
Nature	It is dynamic in nature.	It is static in nature.
Resizing	One can easily resize the allocated memory of a pointer later at any given time.	Once we declare the size of an array, we cannot resize it any time we want according to our requirements.
Type of Storage	A typical pointer variable is capable of storing only a single variable within.	The size of any given array decides the total number of variables that it can store within.

- Pointer to function

In Functions Pointers, function's name can be used to get function's address.
A function can also be passed as an arguments and can be returned from a function.

Following are some interesting facts about function pointers :

1) Unlike normal pointers, a function pointer points to code, not data. Typically a function pointer stores the start of executable code.

2) Unlike normal pointers, we do not allocate de-allocate memory using function pointers.

3) A function's name can also be used to get functions' address. For example, in the below program, we have removed address operator '&' in assignment. We have also changed function call by removing *, the program still works.

Declaration

function_return_type (*Pointer_name) (function argument list)

Ex:

```
#include<stdio.h>
int subtraction (int a, int b) {
  return a-b;
}
int main() {
  int (*fp) (int, int)=subtraction;
  //Calling function using function pointer
  int result = fp(5, 4);
  printf(" Using function pointer we get the result: %d",result);
  return 0;
}
```

Output

Using function pointer we get the result: 1