VOID POINTERS IN C

In this article we are learning about “void pointers” in C language. Before going further it will be good if you refresh about pointers by reading – **Introduction to pointers in C.** A pointer variable is usually declared with the data type of the “content” that is to be stored inside the memory location (to which the pointer variable points to).

Ex:- char *ptr; int *ptr; float *ptr;

A pointer variable declared using a particular data type can not hold the location address of variables of other data types. It is invalid and will result in a compilation error.

Ex:- char *ptr;
int var1;
ptr=&var1; // This is invalid because ‘ptr’ is a character pointer variable.

Here comes the importance of a **“void pointer”**. A void pointer is nothing but a pointer variable declared using the reserved word in C ‘void’.

Ex:- void *ptr; // Now ptr is a general purpose pointer variable

When a pointer variable is declared using keyword void – it becomes a general purpose pointer variable. Address of any variable of any data type (char, int, float etc.)can be assigned to a void pointer variable.

Dereferencing a void pointer

We have seen about dereferencing a pointer variable in our article – Introduction to pointers in C. We use the indirection operator * to serve the purpose. But in the case of a void pointer we need to typecast the pointer variable to dereference it. This is because a void pointer has no data type associated with it. There is no way the compiler can know (or guess?) what type of data is pointed to by the void pointer. So to take the data pointed to by a void pointer we typecast it with the correct type of the data holded inside the void pointers location.

Example program:-

```
#include
void main()
{
int a=10;
```

```
float b=35.75;
void *ptr; // Declaring a void pointer
ptr=&a; // Assigning address of integer to void pointer.
printf("The value of integer variable is= %d",*( (int*) ptr) );// (int*)ptr -
is used for type casting. Where as *((int*)ptr) dereferences the typecasted
void pointer variable.
ptr=&b; // Assigning address of float to void pointer.
printf("The value of float variable is= %f",*( (float*) ptr) );
}
```

The output:-

The value of integer variable is= 10
The value of float variable is= 37.75

A void pointer can be really useful if the programmer is not sure about the data type of data inputted by the end user. In such a case the programmer can use a void pointer to point to the location of the unknown data type. The program can be set in such a way to ask the user to inform the type of data and type casting can be performed according to the information inputted by the user. A code snippet is given below.

```
void funct(void *a, int z)
{
if(z==1)
printf("%d",*(int*)a); // If user inputs 1, then he means the data is an
integer and type casting is done accordingly.
else if(z==2)
printf("%c",*(char*)a); // Typecasting for character pointer.
else if(z==3)
printf("%f",*(float*)a); // Typecasting for float pointer
}
```

Another important point you should keep in mind about void pointers is that – pointer arithmetic can not be performed in a void pointer.

Example:-

```
void *ptr;
int a;
ptr=&a;
ptr++; // This statement is invalid and will result in an error because 'ptr'
is a void pointer variable.
```

Source : http://www.circuitstoday.com/void-pointers-in-c