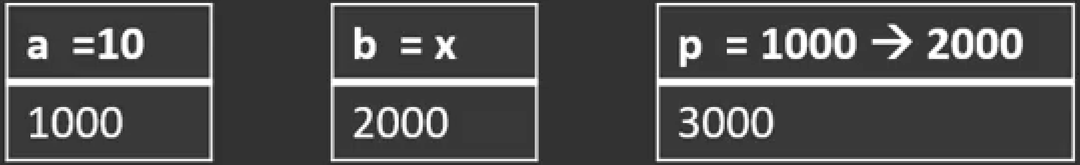
Program ...

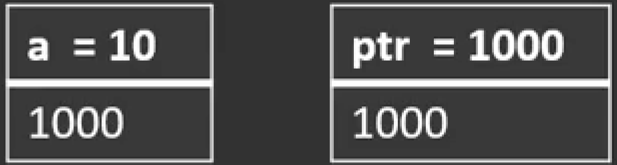
```
int a = 10;
char b = 'x';

void *p = &a;  // void pointer holds address of int 'a'
p = &b; // void pointer holds address of char 'b'
```



Note: void pointer can contain the address of a variable of any type.

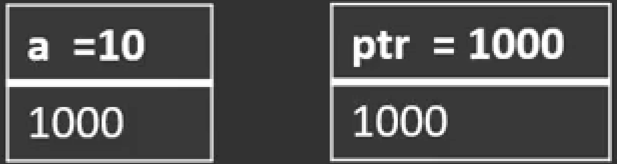
Program ...

```
#include<stdio.h>
int main()
{
    int a = 10;
    void *ptr = &a;
    printf("%d", *ptr);
    return 0;
}
```



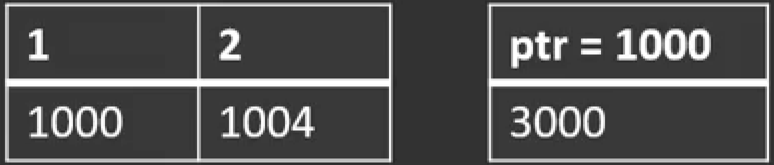
While dereferencing ptr, how compiler will get to know that how many bytes it has to fetch from this address, as it is void pointer? Hence throw an error.

```
#include<stdio.h>
int main()
{
    int a = 10;
    void *ptr = &a;
    printf("%d", *(int *)ptr);
    return 0;
}
```



While dereferencing ptr, now compiler knows that it has to fetch bytes equal to sizeof(int), hence it will give 10 as value.

```
#include<stdio.h>
int main()
{
    int a[2] = {1, 2};
    void *ptr = &a;
    ptr = ptr + sizeof(int);
    printf("%d", *(int *)ptr);
    return 0;
}
```



Ptr = 1000 → 1004 → typecasting to int pointer → 2

Advantages of void pointers:

1) malloc() and calloc() return void * type and this allows these functions to be used to allocate memory of any data type (just because of void *).

2) void pointers in C are used to implement generic functions in C. For example compare function used in qsort.

Thank you for watching!

Please leave us your comments.