We have learned in chapter Pointer Basics in C that if a pointer is of type pointer to `int` or `(int *)` then it can hold the address of the variable of type `int` only. It would be incorrect, if we assign an address of a `float` variable to a pointer of type pointer to `int`. But `void` pointer is an exception to this rule. A `void` pointer can point to a variable of any data type. Here is the syntax of `void` pointer.

```
void *vp;
```

Let's take an example:

```
void *vp;

int a = 100, *ip;
float f = 12.2, *fp;
char ch = 'a';
```

Here `vp` is a `void` pointer, so you can assign the address of any type of variable to it.

```
vp = &a; // ok
vp = ip; // ok
vp = fp; // ok

ip = &f; // wrong since type of ip is pointer to int
fp = ip; // wrong since type of fp is pointer to float
```

A `void` pointer can point to a variable of any data type and `void` pointer can be assigned to a pointer of any type.

We can't just dereference a void pointer using indirection (`*`) operator. For example:

```
void *vp;
```

```
int a = 100;

vp = &a;
printf("%d", *vp); // wrong
```

It simply doesn't work that way!. Before you dereference a void pointer it must be typecasted to appropriate pointer type. Let me show you what I mean.

For example: In the above snippet `void` pointer `vp` is pointing to the address of integer variable a. So in this case `vp` is acting as a pointer to `int` or `(int *)`. Hence the proper typecast in this case is `(int*)`.

```
(int *)vptr
```

Now the type of `vptr` temporarily changes from `void` pointer to pointer to `int` or `(int*)` , and we already know how to dereference a pointer to `int`, just precede it with indirection operator (`*`)

```
*(int *)vptr
```

typecasting changes type of `vp` temporarily until the evaluation of the expression, everywhere else in the program `vp` is still a void pointer.

The following program demonstrates how to dereference a `void` pointer.

```
#include<stdio.h>
#define SIZE 10

int main()
{
    int i = 10;
    float f = 2.34;
    char ch = 'k';

    void *vptr;

    vptr = &i;
    printf("Value of i = %d\n", *(int *)vptr);

    vptr = &f;
    printf("Value of f = %.2f\n", *(float *)vptr);

    vptr = &ch;
    printf("Value of ch = %c\n", *(char *)vptr);

```

```
    // signal to operating system program ran fine
    return 0;
}
```

```
Value of i = 10
Value of f = 2.34
Value of ch = k
```

Another important point I want to mention is about pointer arithmetic with void pointer. Before you apply pointer arithmetic in void pointers make sure to provide a proper typecast first otherwise you may get unexcepted results.

Consider the following example:

```
int one_d[5] = {12, 19, 25, 34, 46}, i;
void *vp = one_d;

printf("%d", one_d + 1); // wrong
```

Here we have assigned the name of the array `one_d` to the void pointer `vp`. Since the base type of `one_d` is a pointer to `int` or `(int*)`, the void pointer `vp` is acting like a pointer to `int` or `(int*)`. So the proper typecast is `(int*)`.

```
int one_d[5] = {12, 19, 25, 34, 46}, i;
void *vp = one_d;

printf("%d", (int *)one_d + 1); // correct
```

The following program demonstrates pointer arithmetic in void pointers.

```
#include<stdio.h>
#define SIZE 10

int main()

```

```
{
    int one_d[5] = {12, 19, 25, 34, 46}, i;

    void *vp = one_d;

    for(i = 0; i < 5; i++)
    {
        printf("one_d[%d] = %d\n", i, *( (int *)vp + i ) );
    }

    // signal to operating system program ran fine
    return 0;
}
```

```
one_d[0] = 12
one_d[1] = 19
one_d[2] = 25
one_d[3] = 34
one_d[4] = 46
```

The void pointers are used extensively in dynamic memory allocation which we will discuss next.

Type Comment Here (at least 3 chars)

Name E-mail Website (optional) Submit

• 10 months ago

a few typo in snippets under "Pointer Arithmetic in Void pointers":
int one_d[5] = {12, 19, 25, 34, 46}, i;

```
void *vp = one_d;

printf("%d", arr + 1); // wrong => printf("%d", one_d+ 1);
```

Same issue, for the next snippet.

-3 |

• 4 months ago

Thanks for posting! :)

-1 |

• 3 weeks ago

Nice article on the void pointer.

 |

Type Comment Here (at least 3 chars)

Name E-mail Website (optional) Submit