



(<https://www.educba.com/software-development/>)

← (<https://www.educba.com/const-pointer-in-c/>)

→ (<https://www.educba.com/pointer-arithmetic-in-c/>)

Dangling Pointers in C

```
#include <stdlib.h>
#include <stdio.h>
int main()
{
    int *ptr1 = (int *)malloc(sizeof(int));
    free(ptr1);
    ptr1 = NULL;
}
```

Output:

```
C:\Users\PAUHAN\Desktop\test\c test>gcc "new 1.c"
C:\Users\PAUHAN\Desktop\test\c test>"C:\Users\PAUHAN\Desktop\test\c test\1.exe"
C:\Users\PAUHAN\Desktop\test\c test>
```

www.educba.com

Introduction to Dangling Pointers in C

The C Dangling pointer is a type of pointer that actually points to a specific memory location that is to be free or deleted. There are some different ways where the pointer now acts as a dangling pointer. Most of the times there are only 3 different types/ways where the pointer will





(<https://www.educba.com/software-development/>)

modifying pointer value. In this topic, we are going to learn about Dangling Pointers in C.

Syntax:

Start Your Free Software Development Course

Web development, programming languages, Software testing & others

```
free(a1)
```

How Dangling Pointers Works in C?

The dangling pointers are similar pointer just like the normal pointer but it works by taking consideration of a de-allocated object/deleted object. It is nothing but a pointer which actually going to point a specific memory location that is actually deleted and it is called a dangling pointer.

The dangling pointer's errors can only be avoided just by initializing the pointer to one NULL value. If we try assigning the NULL value to a specific pointer, then that pointer will not at all point to the needed deallocated memory. Assigning the NULL value to the specific pointer helps the pointer not pointing to any specific memory location.

For de-allocating memory of the C dangling pointer concept, `free()` function is used with single parameter just to make a pointer into a dangling pointer. This is how the dangling pointer will be created with `free()` function in the C coding language. There is also another way of





(<https://www.educba.com/software-development/>)

some value to the specific pointer then it will overwrite the value of the program code.

Examples of Dangling Pointers in C

Here are the following examples mention below:

Example #1

This is the example of the de-allocation of the memory of the C Programming Language by the specific ptr causes. At first, standard libraries or C language is included with the #include method/function. Then int main() is created to write C Code. The *ptr is created to get a pointer variable which is with the help of the malloc() function. The malloc() function usually returns the void value, so here int * is used to convert the void pointer to an int pointer. Then free() function with the parameter "ptr1" is used in order to make the pointer as a dangling pointer. So when the compiler completed the execution, the compiler will run the code perfectly but there will be no output because nothing is mentioned too in print to show as output in the command prompt.

🔗 Popular Course in this category



C Programming Training (3 Courses, 5 Project)

3 Online Courses | 5 Hands-on Projects | 34+ Hours | Verifiable Certificate of Completion | Lifetime Access

★★★★★ 4.5 (8,589 ratings)

Course Price

\$79 ~~\$399~~

View Course





[\(https://www.educba.com/software-development/\)](https://www.educba.com/software-development/)

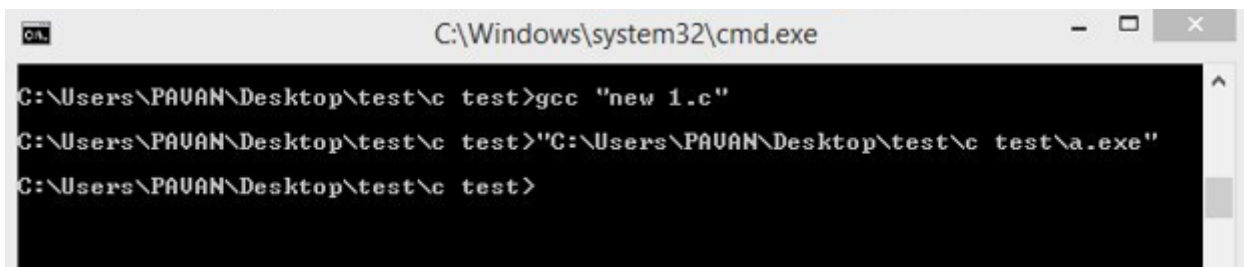
development/courses/c-course/?btnz=edu-blg-inline-banner1)

Java Training (40 Courses, 29 Projects, 4 Quizzes) (<https://www.educba.com/software-development/courses/java-course/?btnz=edu-blg-inline-banner1>)

Syntax:

```
#include <stdlib.h>
#include <stdio.h>
int main()
{
    int *ptr1 = (int *)malloc(sizeof(int));
    free(ptr1);
    ptr1 = NULL;
}
```

Output:



Example #2



This is the example of implementing the Function call way or representing the dangling pointer. Here one of the pointers which point to the local variable becomes into a dangling pointer when



[\(https://www.educba.com/software-development/\)](https://www.educba.com/software-development/)

points to the one which is not at all a valid one anymore. Then printf is used to print. But here there will be a warning when the c code runs in the c compiler. Check out the output so that you will know. Here in this example, the normal pointer doesn't even become into a dangling pointer.

Syntax:

```
#include<stdio.h>

int *fun1()
{
    int x1 = 5;
    return &x1;
}

int main()
{
    int *p1 = fun1();
    fflush(stdin);
    printf("%d", *p1);
    return 0;
}
```

Output:

```
C:\Windows\system32\cmd.exe

C:\Users\PAVAN\Desktop\test\c test>gcc "new 1.c"
new 1.c: In function 'fun1':
new 1.c:7:12: warning: function returns address of local variable [-Wreturn-local-addr]
    return &x1;
```





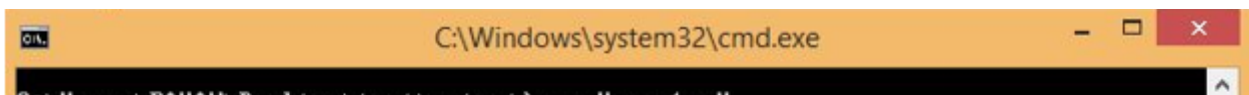
[\(https://www.educba.com/software-development/\)](https://www.educba.com/software-development/)

This is also an example of the function call which is similar to the above example. At first, usually #include is used for including the standard library. Then a function fun11() is created and included a static int variable "x11" with a value "5". Then the main() function is used along with the pointer variable "P11" to include the pointer function fun11(). Then fflush() function is used. The fflush function is mostly used for output streams. Fflush(stdin) is a type of undefined behavior. Then printf() function is used to print the pointer variable which is nothing but the x11 variable value.

Syntax:

```
#include<stdio.h>
int *fun11()
{
    static int x11 = 5;
    return &x11;
}
int main()
{
    int *p11 = fun11();
    fflush(stdin);
    printf("%d", *p11);
    return 0;
}
```

Output:





(<https://www.educba.com/software-development/>)

This is the example of implementing the variable which goes out of the scope. Here the variable will go out of the scope then the pointer pointing to the variable becomes into a dangling pointer. Here at first, we are declaring the pointer variable str1. Then inside we are declaring a character variable. Now the str1 variable contains the variable "a1"s address. Then control will come out of inner scope. Here a1 variable will no longer available. So str1 will point to a specific deallocated memory. It means the str1 pointer will become into a dangling pointer but A1 is an undeclared variable.

Syntax:

```
#include<stdio.h>
int main()
{
    char *str1;
    {
        char a1 = ?A1?;
        str1 = &a1;
    }
    printf("%s", *str1);
}
```

Output:

```
C:\Windows\system32\cmd.exe
C:\Users\PAUWAN\Desktop\test\c test>gcc "new 1.c"
new 1.c: In function 'main':
new 1.c:6:19: error: expected expression before '?' token
        char a1 = ?A1?;
                   ^
```



[\(https://www.educba.com/software-development/\)](https://www.educba.com/software-development/)

CONCLUSION

I hope you understand what is the definition of C Dangling/Wild Pointers along with its syntax and explanation, how the dangling pointers work in C Programming Language along with various examples of implementing better and so easily.

Recommended Articles

This is a guide to Dangling Pointers in C. Here we discuss how Dangling Pointers Works in C along with programming examples to understand better. You may also have a look at the following articles to learn more –

1. [Patterns in C Programming \(https://www.educba.com/patterns-in-c-programming/\)](https://www.educba.com/patterns-in-c-programming/)
2. [C Literals \(https://www.educba.com/c-literals/\)](https://www.educba.com/c-literals/)
3. [C Programming Matrix Multiplication \(https://www.educba.com/c-programming-matrix-multiplication/\)](https://www.educba.com/c-programming-matrix-multiplication/)

ALL IN ONE SOFTWARE DEVELOPMENT BUNDLE (600+ COURSES, 50+ PROJECTS)

- ☒ 600+ Online Courses
- ☒ 50+ projects
- ☒ 3000+ Hours
- ☒ Verifiable Certificates
- ☒ Lifetime Access

Learn More

<https://www.educba.com/software-development/courses/software-development-course/?btnz=edu-blg-inline-banner3>





[\(https://www.educba.com/software-development/\)](https://www.educba.com/software-development/)

About Us

Blog (<https://www.educba.com/blog/?source=footer>)

Who is EDUCBA? (<https://www.educba.com/about-us/?source=footer>)

Sign Up (<https://www.educba.com/software-development/signup/?source=footer>)

Corporate Training (<https://www.educba.com/corporate/?source=footer>)

Certificate from Top Institutions (<https://www.educba.com/educbalive/?source=footer>)

Contact Us (<https://www.educba.com/contact-us/?source=footer>)

Verifiable Certificate (<https://www.educba.com/software-development/verifiable-certificate/?source=footer>)

Reviews (<https://www.educba.com/software-development/reviews/?source=footer>)

Terms and Conditions (<https://www.educba.com/terms-and-conditions/?source=footer>)

Privacy Policy (<https://www.educba.com/privacy-policy/?source=footer>)

Apps

iPhone & iPad (<https://itunes.apple.com/in/app/educba-learning-app/id1341654580?mt=8>)

Android (<https://play.google.com/store/apps/details?id=com.educba.www>)





[\(https://www.educba.com/software-development/\)](https://www.educba.com/software-development/)

Python Tutorials (<https://www.educba.com/software-development/software-development-tutorials/python-tutorial/?source=footer>)

All Tutorials (<https://www.educba.com/software-development/software-development-tutorials/?source=footer>)

Certification Courses

All Courses (<https://www.educba.com/software-development/courses/?source=footer>)

Software Development Course - All in One Bundle (<https://www.educba.com/software-development/courses/software-development-course/?source=footer>)

Become a Python Developer (<https://www.educba.com/software-development/courses/python-certification-course/?source=footer>)

Java Course (<https://www.educba.com/software-development/courses/java-course/?source=footer>)

Become a Selenium Automation Tester (<https://www.educba.com/software-development/courses/selenium-training-certification/?source=footer>)

Become an IoT Developer (<https://www.educba.com/software-development/courses/iot-course/?source=footer>)

ASP.NET Course (<https://www.educba.com/software-development/courses/asp-net-course/?source=footer>)

VB.NET Course (<https://www.educba.com/software-development/courses/vb-net-course/?source=footer>)

PHP Course (<https://www.educba.com/software-development/courses/php-course/?source=footer>)

© 2020 - EDUCBA. ALL RIGHTS RESERVED. THE CERTIFICATION NAMES ARE THE TRADEMARKS OF THEIR RESPECTIVE OWNERS.

