Output of the program | Dereference, Reference, Dereference, Reference....
Dangling, Void , Null and Wild Pointers
An Uncommon representation of array elements
How to declare a pointer to a function?
Pointer vs Array in C
void pointer in C / C++
NULL pointer in C
Function Pointer in C
Difference between pointer to an array and array of pointers
vowels | SET 2
C program to sort an array using pointers
Basic Code Optimizations in C
dot (.) operator in C/C++
Features and Use of Pointers in C/C++
How can we use Comma operator in place of curly braces?
Difference between while and do-while loop in C, C++, Java
Sum of an array using MPI
__builtin_inf() functions of GCC compiler
C Program to count the Number of Characters in a File
time.h header file in C with Examples
scanf("%[^\n]s", str) Vs gets(str) in C with Examples
AKTU (UPTU) Previous Year Solved Papers | C Programming
Constants vs Variables in C language
Analyzing BufferOverflow with GDB
C program to Insert an element in an Array
Types of Literals in C/C++ with Examples
Conditional or Ternary Operator (?:) in C/C++
Difference between C and C#
time.h localtime() function in C with Examples

Dangling, Void , Null and Wild Pointers

Dangling pointer

A pointer pointing to a memory location that has been deleted (or freed) is called dangling pointer. There are **three** different ways where Pointer acts as dangling pointer

1. **De-allocation of memory**

```
// Deallocating a memory pointed by ptr causes
#include <stdlib.h>
#include <stdio.h>
int main()
{
    int *ptr = (int *)malloc(sizeof(int));

    // After below free call, ptr becomes a
    // dangling pointer
    free(ptr);

    // No more a dangling pointer
    ptr = NULL;
}
```

2. **Function Call**

```
// The pointer pointing to local variable becomes
// dangling when local variable is not static.
#include<stdio.h>

int *fun()
{
    // x is local variable and goes out of
    // scope after an execution of fun() is
    // over.
    int x = 5;

    return &x;
}

// Driver Code
int main()
{
    int *p = fun();
    fflush(stdin);

    // p points to something which is not
    // valid anymore
    printf("%d", *p);
    return 0;
}
```

Output:

```
A garbage Address
```

The above problem doesn't appear (or p doesn't become dangling) if x is a static variable.

```
// The pointer pointing to local variable doesn't
// become dangling when local variable is static.
#include<stdio.h>

int *fun()
{
    // x now has scope throughout the program
    static int x = 5;

    return &x;
}

int main()
{
    int *p = fun();
    fflush(stdin);

    // Not a dangling pointer as it points
    // to static variable.
    printf("%d",*p);
}
```

Output:

```
5
```

3. **Variable goes out of scope**

```
void main()
{
   int *ptr;
   .....
   .....
   {
       int ch;
       ptr = &ch;
   }
   .....
   // Here ptr is dangling pointer
}
```

Void pointer

Void pointer is a specific pointer type – void * – a pointer that points to some data location in storage, which doesn't have any specific type. Void refers to the type. Basically the type of data that it points to is can be any. If we assign address of char data type to void pointer it will become char Pointer, if int data type then int pointer and so on. Any pointer type is convertible to a void pointer hence it can point to any value.

Important Points

1. void pointers **cannot be dereferenced**. It can however be done using typecasting the void pointer
2. Pointer arithmetic is not possible on pointers of void due to lack of concrete value and thus size.

Example:

```
#include<stdlib.h>

int main()
{
    int x = 4;
    float y = 5.5;

    //A void pointer
    void *ptr;
    ptr = &x;

    // (int*)ptr - does type casting of void
    // *((int*)ptr) dereferences the typecasted
    // void pointer variable.
    printf("Integer variable is = %d", *( (int*) ptr) );

    // void pointer is now float
    ptr = &y;
    printf("\nFloat variable is= %f", *( (float*) ptr) );

    return 0;
}
```

Output:

```
Integer variable is = 4
Float variable is= 5.500000
```

Refer void pointer article for details.

NULL Pointer

NULL Pointer is a pointer which is pointing to nothing. In case, if we don't have address to be assigned to a pointer, then we can simply use NULL.

```
#include <stdio.h>
int main()
{
    // Null Pointer
    int *ptr = NULL;

    printf("The value of ptr is %p", ptr);
    return 0;
}
```

Output :

```
The value of ptr is (nil)
```

Important Points

1. **NULL vs Uninitialized pointer –** An uninitialized pointer stores an undefined value. A null pointer stores a defined value, but one that is defined by the environment to not be a valid address for any member or object.
2. **NULL vs Void Pointer** – Null pointer is a value, while void pointer is a type

Wild pointer

A pointer which has not been initialized to anything (not even NULL) is known as wild pointer. The pointer may be initialized to a non-NULL garbage value that may not be a valid address.

```
int main()
{
    int *p;  /* wild pointer */

    int x = 10;

    // p is not a wild pointer now
    p = &x;

    return 0;
}
```

This article is contributed by **Spoorthi Arun**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recommended Posts:

What are Wild Pointers? How can we avoid?
void pointer in C / C++
How does "void *" differ in C and C++?
Return from void functions in C++
NULL pointer in C
Passing NULL to printf in C
Understanding "static" in "public static void main" in Java
Is it fine to write "void main()" or "main()" in C/C++?
Difference between "int main()" and "int main(void)" in C/C++?
Applications of Pointers in C/C++
Pointers and References in C++
Features and Use of Pointers in C/C++
Smart Pointers in C++
Pointers in C/C++ with Examples
What are near, far and huge pointers?

Improved By : ashoklb, sharathmaidargi

Article Tags : C C++ School Programming C-Pointers cpp-pointer
Practice Tags : C CPP

52

2.2
Based on **101** vote(s)

To-do Done

Feedback/ Suggest Improvement Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Previous
Execution of printf with ++ operators
Next
Default array values in Java

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Most popular in C
Arrow operator -> in C/C++ with Examples
Logical Not ! operator in C with Examples
size of char datatype and char array in C
return statement in C/C++ with Examples
asctime() and asctime_s() functions in C with Examples

Most visited in C++
std::uniform_real_distribution b() method in C++ with Examples
std::uniform_real_distribution max() method in C++ with Examples
std::uniform_real_distribution a() method in C++ with Examples
std::uniform_real_distribution reset() method in C++ with Examples
Weighted K-NN