

# Pointers

|                |                           |
|----------------|---------------------------|
| ☰ Tags         | c pointers                |
| 🕒 Created time | @October 2, 2024 11:56 PM |
| # Lecture No.  | 7                         |
| ☑ Reviewed     | <input type="checkbox"/>  |

## Why do we need to use pointers?

```
# Example: Write a function to double an input value
int doubleX(int x)
{
    return x*2;
}

int x = 5;
x = doubleX(x);
```

The problem of the above approach is the amount of used memory and complexity. In particular, the value of `x` is copied and passed to the function `doubleX`. The returned value of `doubleX` is then assigned to `x`. Is there any other way to do this task? With pointers!

## Intro to Pointers

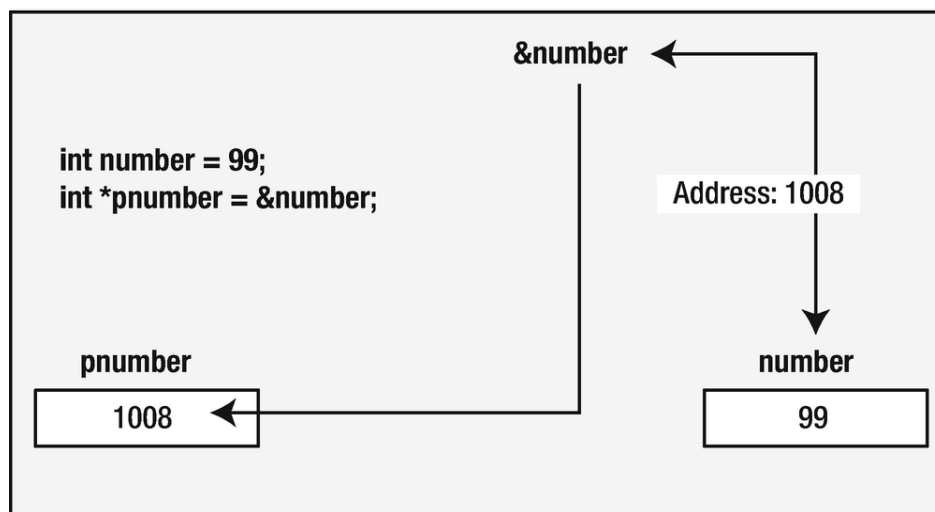
A pointer is a variable that stores the memory address of another variable. Normally when you create a variable it holds a value e.g

```
int x = 10; // x stores the value 10
```

But sometimes you want to work with the memory location where that value is stored. This is where pointers come in. A pointer doesn't store an actual value like `x` does. Instead it stores the address of where `x` is stored in memory.

For the above `x`:

- An area of memory is allocated to store an integer (10), which can be accessed using the variable name `number`
- The computer references the area or memory using an address.
- Variables that can store addresses are called pointers.
- The address that's stored in a pointer is usually that of another variable as illustrated below:



- In the figure above, there is a pointer called `pnumber` that contains the address of another variable called `number`
- The value that is stored in `pnumber` is the address of the first byte of `number`
- Every pointer will be associated with a specific variable type, and it can only be used to point to variables of that type. For example `int *pnumber` can only point to variables of type `int`
- **Note** a pointer of type `void*` can contain the address of a data item of any type.

# Declaring Pointers

You can declare a pointer as follows:

```
type *pointerName;
```

**Note** you should initialise a declared pointer so that it does not point to anything:

```
int* pnumber = NULL;
```

If you want to initialise your variable `pnumber` with the address of a variable you have already declared, you can use the address of operator `&`

```
int number = 15;
int *pointer = &number;
```

---

## Lecture Break

### Array and Pointers

- An array is a collectin of objects of the type that you can refer to using a single name.
- A pointer holds the address of different variables at different times, as long as they are all of the same type.

```
#include <stdio.h>

int main(int argc, char const *argv[])
{
    int inputArray[3] = {1,2,3};
    int* pFirstElement = &inputArray[0];
```

```

printf("The address of the first element is: %p\n", pFirstE
printf("The address obtained from the array name inputArray
// you can get the value using the * operator
printf("The value of the first element is %d\n", *pFirstEler

printf("The address of the second element is: %p\n", pFirstE
printf("The value of the second element is %d\n", *(pFirstE

printf("The address of the third element is: %p\n", pFirstE
printf("The value of the third element is %d\n", *(pFirstEle

return 0;
}

```

```

% ./array.out
The address of the first element is: 0x7ffffbfa282fc
The address obtained from the array name inputArray is 0x7ffffbfa282fc
The value of the first element is 1
The address of the second element is: 0x7ffffbfa28300
The value of the second element is 2
The address of the third element is: 0x7ffffbfa28304
The value of the third element is 3

```

## Multidimensional Arrays with Pointers

```

#include <stdio.h>

int main(int argc, char const *argv[])
{
    char matrix[3][3] = {
        {'1', '2', '3'},
        {'4', '5', '6'},
        {'7', '8', '9'}
    }
}

```

```

    };
    printf("address of matrix      : %p\n", matrix);
    printf("address of matrix[0][0] : %p\n", &matrix[0][0]);
    printf("value of matrix[0]      : %p\n", matrix[0]);
    return 0;
}

```

```

% ./matrix.out
address of matrix      : 0x7ffe0b23e95f
address of matrix[0][0] : 0x7ffe0b23e95f
value of matrix[0]      : 0x7ffe0b23e95f

```

## String & Pointers

It is possible to declare strings as:

```
char *collegeName = "Dublin City University";
```



**In exams, when stated to use pointers, pointers must be used to complete tasks, otherwise full marks won't be awarded.**

## Searching for a String

From the `string.h` library, the `strchr()` function searches for a given character in the string. The first argument to the function is the string to be searched (which will be the address of a char array), and the second argument is the character that you're looking for.

```
char str[] = "The quick brown fox";    // The string to be searched
char ch = 'q';                        // The character we are looking for
char *pGot_char = NULL;                // Pointer initialized to NULL
pGot_char = strchr(str, ch);           // Stores address where character found

printf("Character found was %c.", *pGot_char);

>> Character found was q.
```

The address of the first character in the string is given by the name of the array. Because 'q' appears as the fifth character in the string, its address will be `str + 4`. Therefore the `pGot_char` will contain this offset address.