



Pointers in C++

Diego Useche - dh.useche@uniandes.edu.co

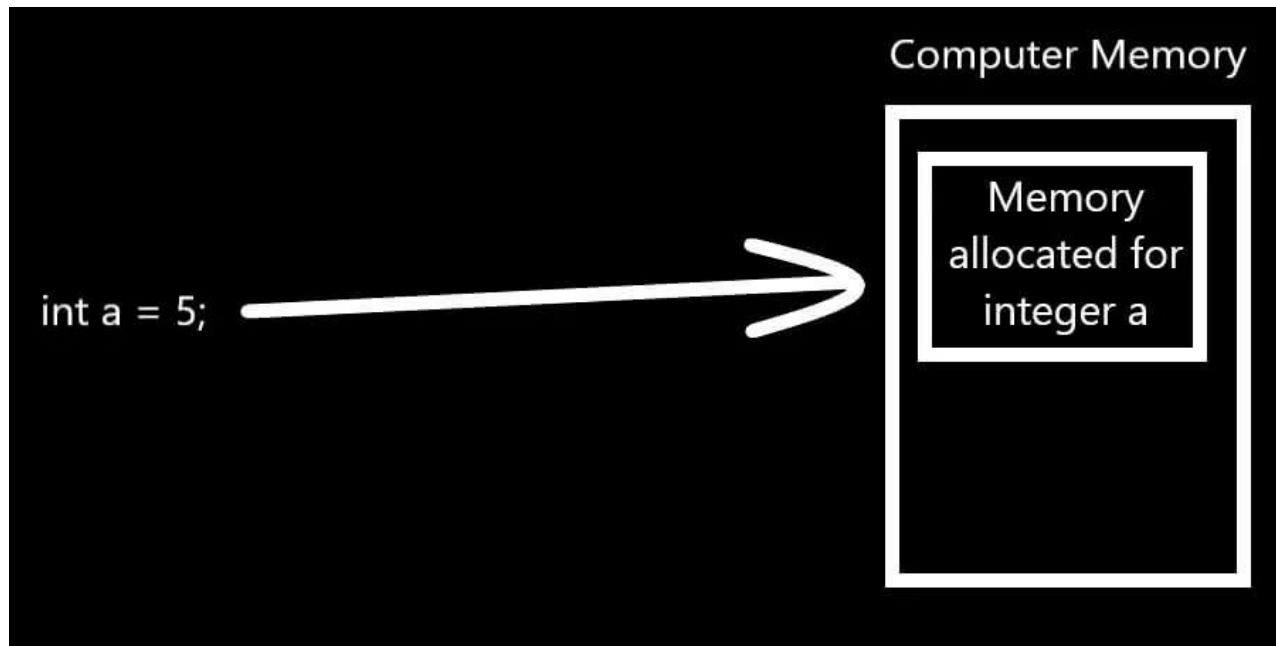
Metodos Computacionales II

Physics Department, Universidad de los Andes, Bogotá



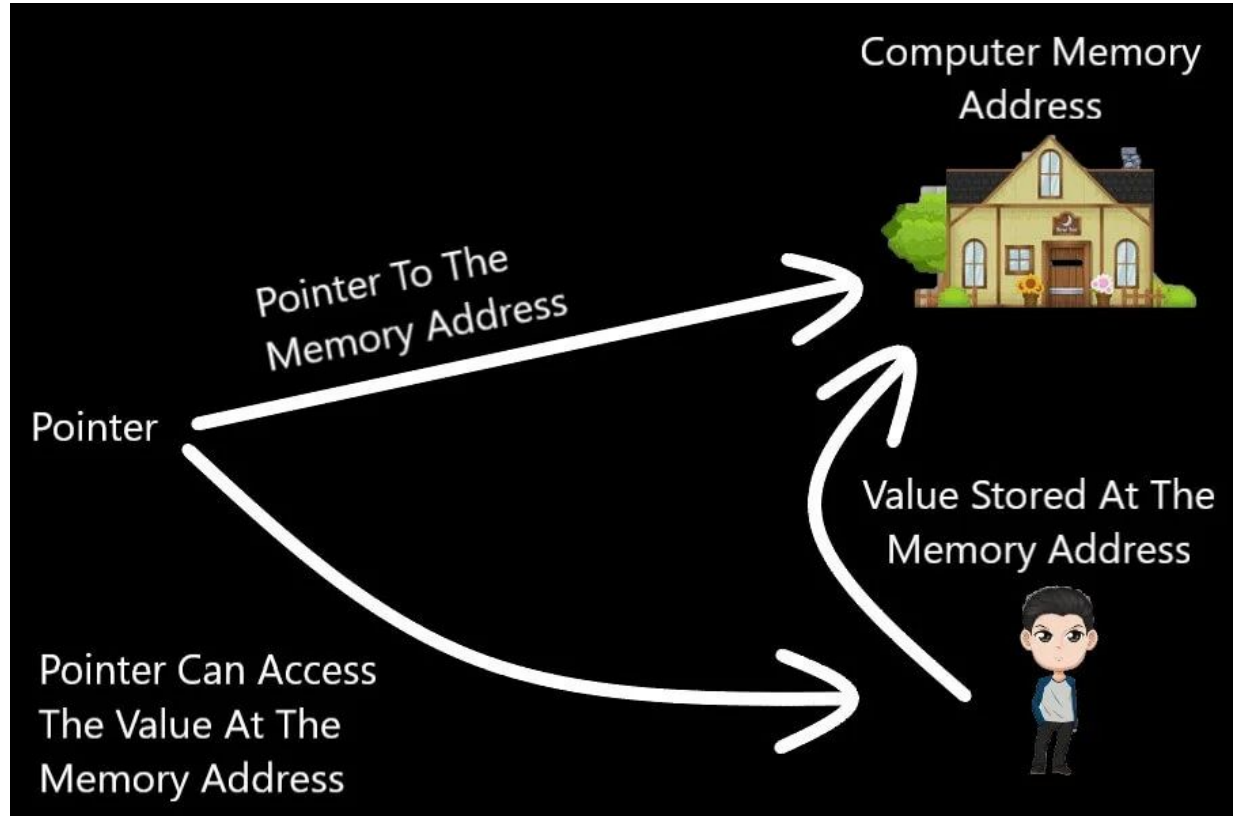
Pointers

- Variables are allocated at some address in memory



Pointers

- A pointer is a variable that indicates the address of another variable



Pointers: pointer address

- `&score`, indicates the address in memory of variable `score`.

```
#include <iostream>
using namespace std;

int main()
{
    int score = 5;
    cout << &score << endl;

    return 0;
}
```

Pointers: pointer address

- `&score`, indicates the address in memory of variable `score`.

```
#include <iostream>
using namespace std;

int main()
{
    int score = 5;
    cout << &score << endl;

    return 0;
}
```

Your Output

0x7fffffa2568bc

prints hexadecimal
address of variable in
memory

Pointers: creating a pointer variable

- a pointer is variable that saves the address in memory of another variable
- `int* scorePtr`, indicates that `scorePtr` is a pointer of an `int`

```
#include <iostream>
using namespace std;

int main()
{
    int score = 5;
    int* scorePtr;
    scorePtr = &score;

    cout << scorePtr << endl;

    return 0;
}
```

Your Output

0x7ffffa2568bc

Pass by reference vs pass by value

By value

```
#include <iostream>
using namespace std;

void myFunc(int x) {
    x = 100;
}

int main() {
    int var = 20;
    myFunc(var);
    cout << var;
}
```

By reference

```
#include <iostream>
using namespace std;

void myFunc(int* x) {
    *x = 100;
}

int main() {
    int var = 20;
    myFunc(&var);
    cout << var;
}
```

Pass by reference vs pass by value

By value

```
#include <iostream>
using namespace std;

void myFunc(int x) {
    x = 100;
}

int main() {
    int var = 20;
    myFunc(var);
    cout << var;
}
```

Result: 20

By reference

```
#include <iostream>
using namespace std;

void myFunc(int* x) {
    *x = 100;
}

int main() {
    int var = 20;
    myFunc(&var);
    cout << var;
}
```

Result: 100

Create a pointer to a Class

We build an instance of the class

```
class Triangle
{
    public:

        float a;
        float b;
        float c;

        Triangle(int a_param, int b_param, int c_param)
        {
            a = a_param;
            b = b_param;
            c = c_param;
        }

        float perimeter()
        {
            float perim = a + b + c;
            return perim;
        }
};

int main() {
    Triangle mytriangle(2, 3, 4);
    cout << mytriangle.a << endl;
    cout << mytriangle.perimeter();
    return 0;
}
```

Create a pointer to a Class

Your Output

```
2
9
```

```
class Triangle
{
    public:

        float a;
        float b;
        float c;

        Triangle(int a_param, int b_param, int c_param)
        {
            a = a_param;
            b = b_param;
            c = c_param;
        }

        float perimeter()
        {
            float perim = a + b + c;
            return perim;
        }
};

int main() {
    Triangle mytriangle(2, 3, 4);
    cout << mytriangle.a << endl;
    cout << mytriangle.perimeter();
    return 0;
}
```

Create a pointer to a Class

Create a pointer to a Class,
and access its members and
functions

```
class Triangle
{
    public:
        float a;
        float b;
        float c;

        Triangle(int a_param, int b_param, int c_param)
        {
            a = a_param;
            b = b_param;
            c = c_param;
        }

        float perimeter()
        {
            float perim = a + b + c;
            return perim;
        }
};

int main() {
    Triangle mytriangle(2, 3, 4);
    Triangle* ptrMytriangle = &mytriangle;
    cout << ptrMytriangle->a << endl;
    cout << ptrMytriangle->perimeter();
    return 0;
}
```

Create a pointer to a Class

Your Output

```
/usercode/file0.cpp: In function 'int main()':
/usercode/file0.cpp:30:24: error: request for member
'a' in 'ptrMytriangle', which is of pointer type
'Triangle*' (maybe you meant to use '->' ?)
 30 |   cout << ptrMytriangle.a << endl;
    |                       ^
/usercode/file0.cpp:31:24: error: request for member
'perimeter' in 'ptrMytriangle', which is of pointer
type 'Triangle*' (maybe you meant to use '->' ?)
 31 |   cout << ptrMytriangle.perimeter();
    |                       ^~~~~~
```

```
class Triangle
{
    public:

    float a;
    float b;
    float c;

    Triangle(int a_param, int b_param, int c_param)
    {
        a = a_param;
        b = b_param;
        c = c_param;
    }

    float perimeter()
    {
        float perim = a + b + c;
        return perim;
    }
};

int main() {
    Triangle mytriangle(2, 3, 4);
    Triangle* ptrMytriangle = &mytriangle;
    cout << ptrMytriangle.a << endl;
    cout << ptrMytriangle.perimeter();
    return 0;
}
```

Create a pointer to a Class

Use " -> " to access the attributes and functions of a pointer of a Class.

```
class Triangle
{
    public:
        float a;
        float b;
        float c;

        Triangle(int a_param, int b_param, int c_param)
        {
            a = a_param;
            b = b_param;
            c = c_param;
        }

        float perimeter()
        {
            float perim = a + b + c;
            return perim;
        }
};

int main() {
    Triangle mytriangle(2, 3, 4);
    Triangle* ptrMytriangle = &mytriangle;
    cout << ptrMytriangle->a << endl;
    cout << ptrMytriangle->perimeter();
    return 0;
}
```

Create a pointer to a Class

Your Output

2
9

```
class Triangle
{
    public:
        float a;
        float b;
        float c;

        Triangle(int a_param, int b_param, int c_param)
        {
            a = a_param;
            b = b_param;
            c = c_param;
        }

        float perimeter()
        {
            float perim = a + b + c;
            return perim;
        }
};

int main() {
    Triangle mytriangle(2, 3, 4);
    Triangle* ptrMytriangle = &mytriangle;
    cout << ptrMytriangle->a << endl;
    cout << ptrMytriangle->perimeter();
    return 0;
}
```

Stack vs Heap Memory

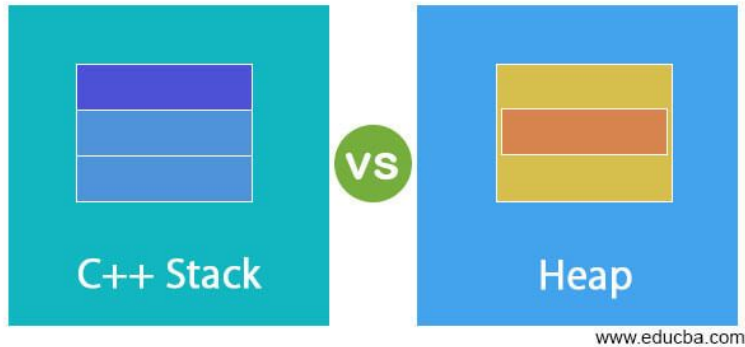
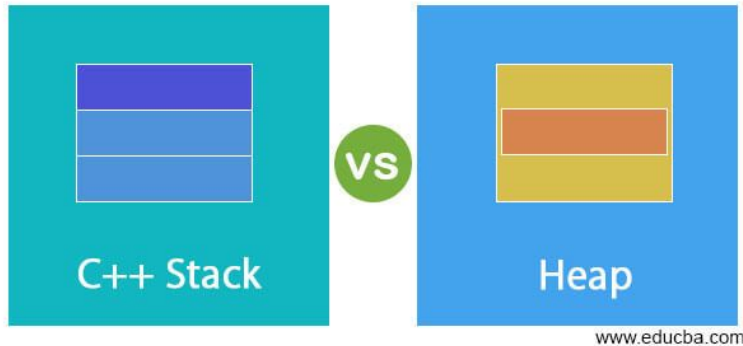


Image taken from <https://www.educba.com/c-stack-vs-heap/>

Stack vs Heap Memory











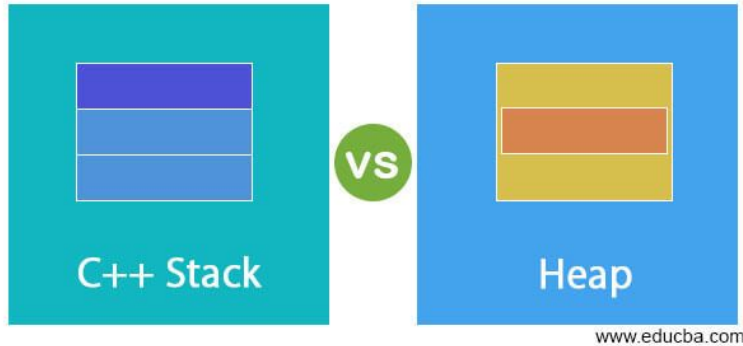








C++ Stack	Heap
 <p>In C++, stack memory is allocated in the contiguous blocks.</p>	 <p>In case of heap, memory is allocated in the computer in random order.</p>
 <p>In terms of accessing the data, stack is comparatively faster than heap.</p>	 <p>Accessing data in heap memory is comparatively slower than stack.</p>
 <p>When it comes to data structure, stack follows the linear data structure.</p>	 <p>Heap in C++ follows the hierarchical data structure.</p>
 <p>In case of stack memory, allocation and de-allocation of memory is done automatically by the compiler.</p>	 <p>In case of heap memory, allocation and deallocation of the memory needs to be done by the programmer programmatically.</p>

Image taken from <https://www.educba.com/c-stack-vs-heap/>

Stack vs Heap Memory



C++ Stack	Heap
 <p>Memory used in stack never gets fragmented as it is efficiently managed by OS at the time of allocation and deallocation.</p>	 <p>Memory used in heap gets fragmented as the blocks of memory first get allocated and then get freed up.</p>
 <p>Stack allows the accessing of local variables only like function, method data, etc.</p>	 <p>Data in the heap can also be accessed globally unlike stack.</p>
 <p>Variables in the stack memory cannot be resized as there is restriction on the memory size.</p>	 <p>In heap, variables can be resized as there is no limit on the memory size.</p>
 <p>Objects in stack memory are automatically destroyed after the function call is finished and the memory is deallocated.</p>	 <p>Programmer needs to explicitly deallocate the memory of the variables in case of heap.</p>

Create a variable in heap

```
#include <iostream>
using namespace std;

int main() {
    int* ptrScore = new int(5);
    cout << *ptrScore << endl;
    cout << ptrScore << endl;
    delete ptrScore;
    return 0;
}
```

use new to create the novel variable in heap

deletions are not handled automatically in heap, pointer variables in heap must always be deleted.

Create a variable in heap

```
#include <iostream>
using namespace std;

int main() {
    int* ptrScore = new int(5);
    cout << *ptrScore << endl;
    cout << ptrScore << endl;
    delete ptrScore;
    return 0;
}
```

use new to create the novel variable in heap

deletions are not handled automatically in heap, pointer values must always be deleted.

Your Output

```
5
0x192ae70
```

Example

Returns

- The function is declared with a void return type, so there is no value to return. Modify the values in memory so that a contains their sum and b contains their absolute difference.
- $a' = a + b$
- $b' = |a - b|$

Input Format

Input will contain two integers, a and b , separated by a newline.

Sample Input

```
4
5
```

Sample Output

```
9
1
```

References

<https://www.cs.mtsu.edu/~xyang/2170/datatypes.html>

<https://www.sololearn.com/>

<https://www.hackerrank.com/>

<https://data-flair.training/blogs/c-tutorial/>

<https://www.astateofdata.com/python-programming/can-python-be-compiled/>

<https://slideplayer.com/slide/15836241/>

<https://www.youtube.com/watch?v=BdnpFbODLc0>

<https://www.educba.com/c-stack-vs-heap/>

<https://awesometuts.com/blog/cpp-pointers/>