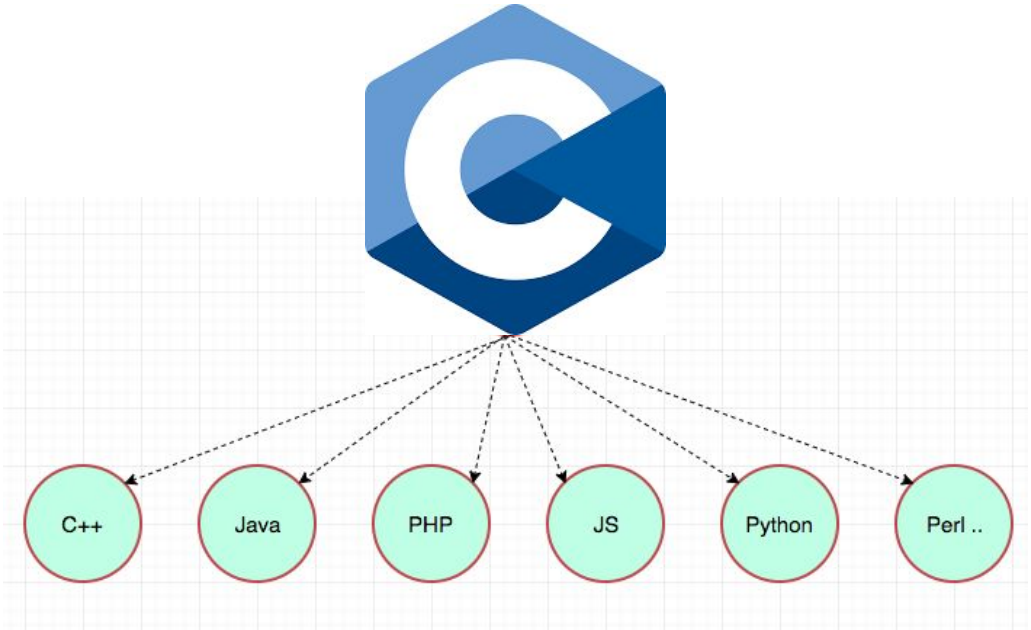
Introducción a C++

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

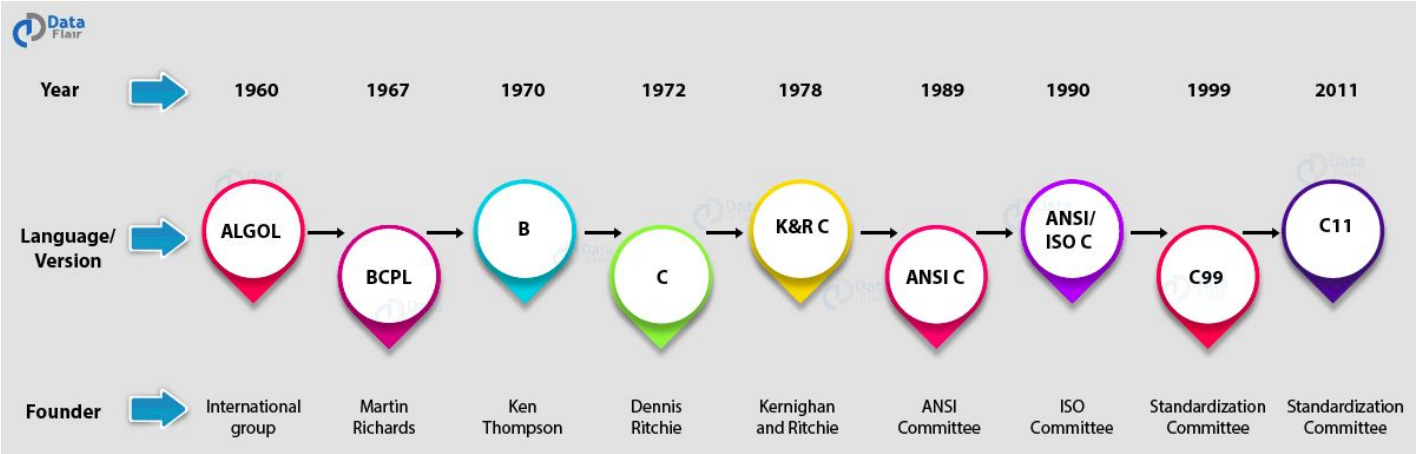
Metodos Computacionales II

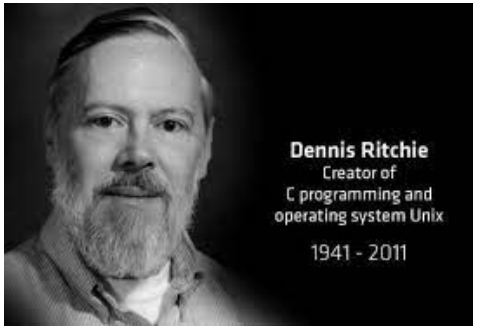
Physics Department, Universidad de los Andes, Bogotá

Origins of C++: Based on C language



Origins of C++: Origins of C language



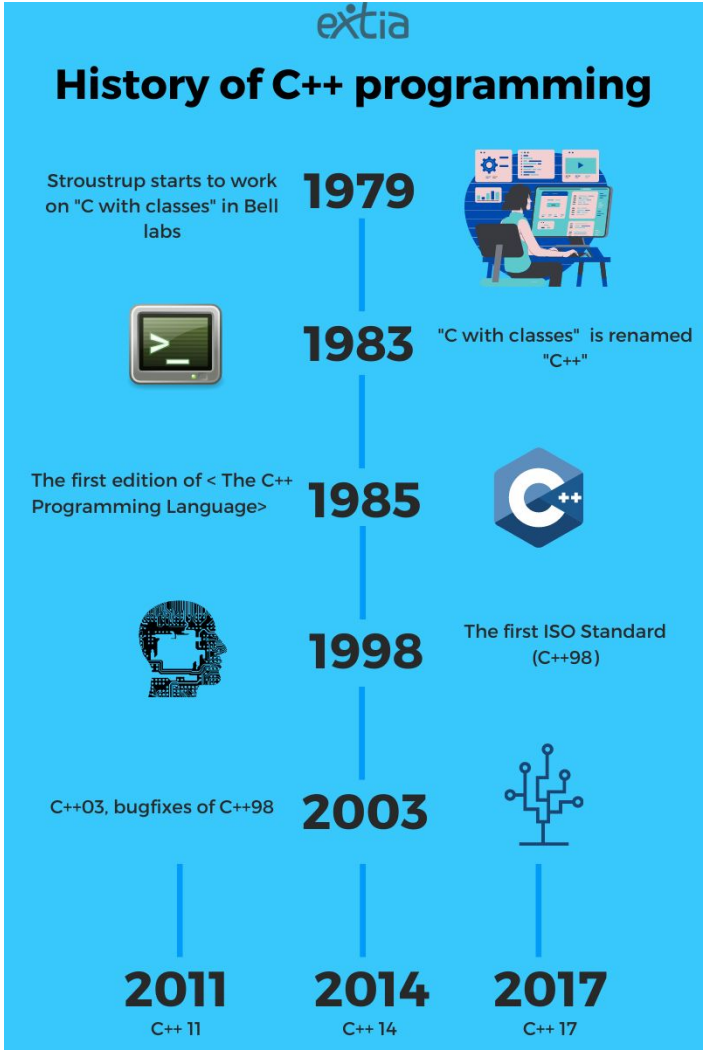


Bell Labs 1970

Photo taken from https://data-flair.training/blogs/c-tutorial/

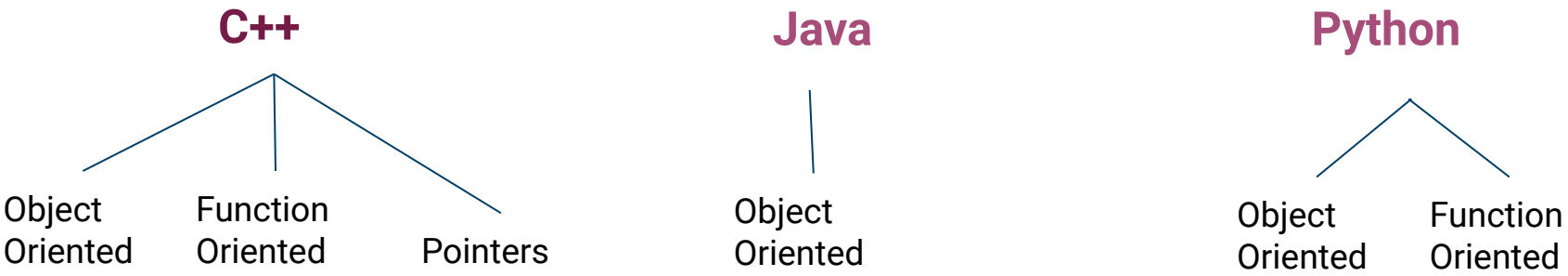
Origins of C++

Bjarne Stroustrup, ATT Labs



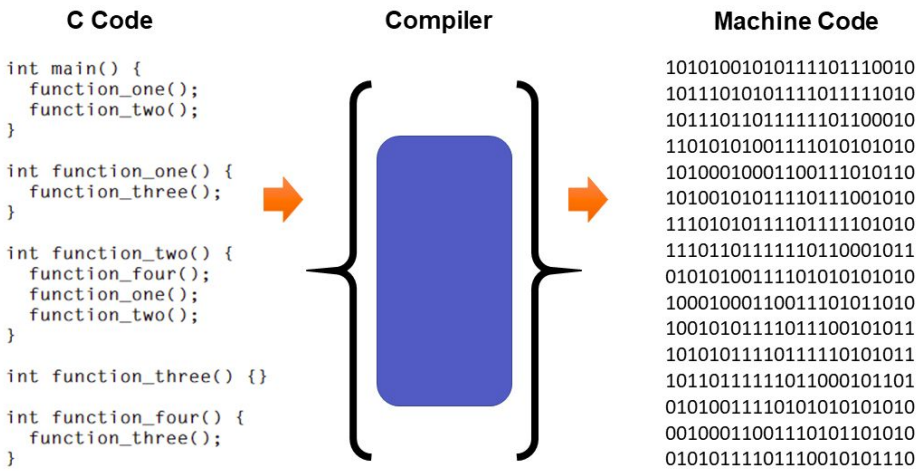
C++ vs Python vs Java

C – printf()
Vs
C++ - cout<<
Vs
Java – println()
Vs
Python – print()



Compiler vs Interpreter language

C++, uses a compiler to convert the whole code to machine language



Python, uses an interpreter to convert line by line code to machine language

```
>>> 3 + 7
10
>>> 3 < 15
True
>>> 'print me'
'print me'
>>> print 'print me'
print me
>>>
```

Images taken from
https://www.astateofdata.com/python-programming/can-python-be-compiled/
https://slideplayer.com/slide/15836241/

Compilation of C++

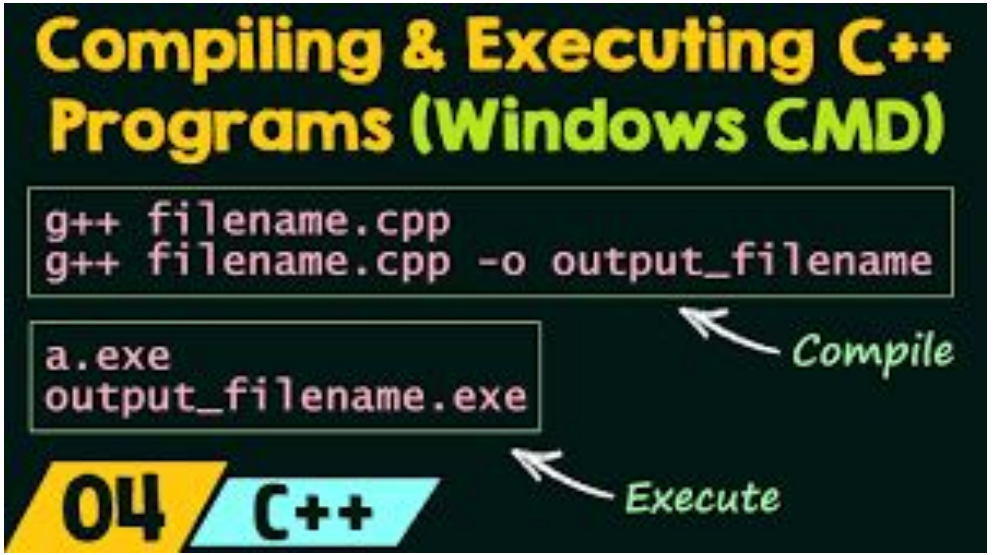


Image taken from https://www.youtube.com/watch?v=BdnpFbODLc0

Initial program

- # to include libraries

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

int main()
{

  return 0;
}
```

Namespace and hello world

- cout without namespace would be std::cout

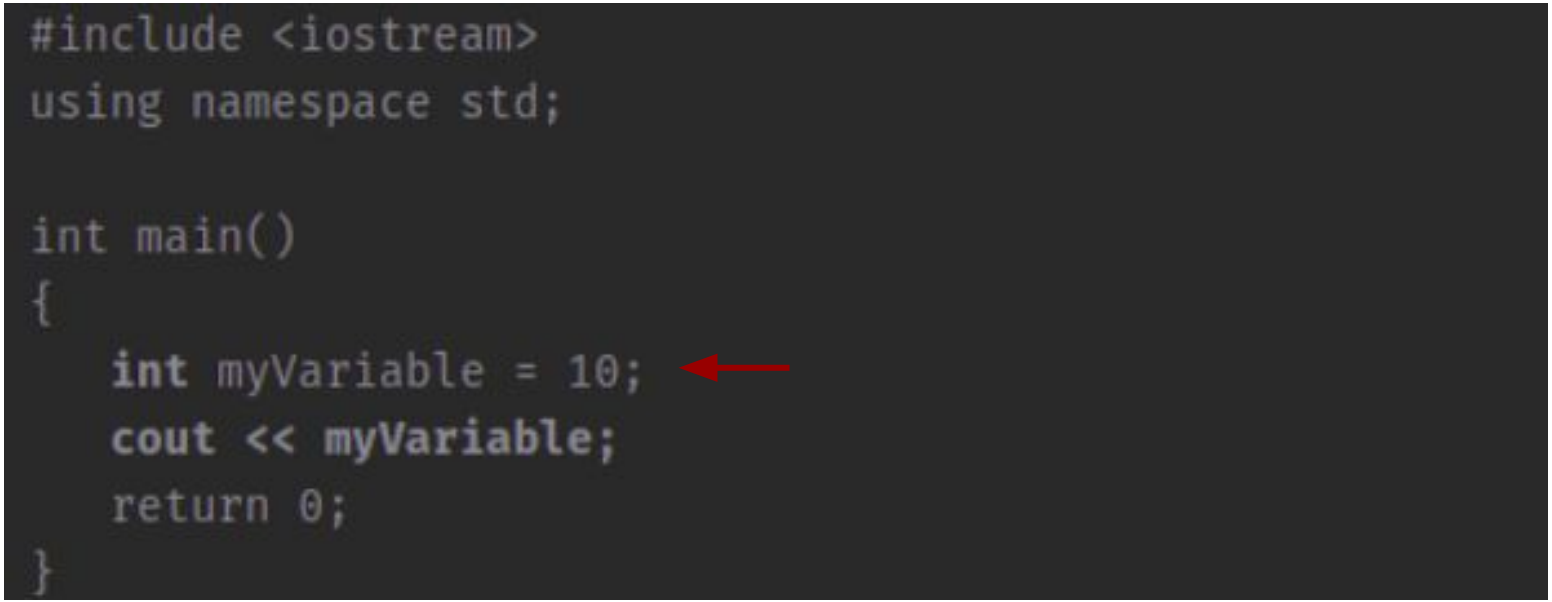
```
#include <iostream>
using namespace std;

int main()
{
cout << "Hello world!";
return 0;
}
```

Comment code

- multiline comment /* */
- single line comment //

```
/*We are going to
    print "Hello world!" */

    cout << "Hello world!"; // prints Hello world!
```

Declaration of variables



Data types

Table 2-6 Integer Data Types

Data Type	Typical Size	Typical Range
`short int`	2 bytes	−32,768 to +32,767
`unsigned short int`	2 bytes	0 to +65,535
`int`	4 bytes	−2,147,483,648 to +2,147,483,647
`unsigned int`	4 bytes	0 to 4,294,967,295
`long int`	4 bytes	−2,147,483,648 to +2,147,483,647
`unsigned long int`	4 bytes	0 to 4,294,967,295
`long long int`	8 bytes	−9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
`unsigned long long int`	8 bytes	0 to 18,446,744,073,709,551,615

Table 2-8 Floating Point Data Types on PCs

Data Type	Key Word	Description
Single precision	`float`	4 bytes. Numbers between ±3.4E-38 and ±3.4E38
Double precision	`double`	8 bytes. Numbers between ±1.7E-308 and ±1.7E308
Long double precision	`long double*`	8 bytes. Numbers between ±1.7E-308 and ±1.7E308

Image taken from https://www.cs.mtsu.edu/~xyang/2170/datatypes.html

Standard input and standard output (input() and print())

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

int main()
{
    int a, b;
    cout << "Enter a number \n";
    cin >> a;
    cout << "Enter another number \n";
    cin >> b;

    return 0;
}
```

Prefix and postfix

```
x = 5;
y = ++x;
// x is 6, y is 6
```

If statement

- put curly brackets { }

```
int a = 55;
int b = 33;
if (a > b) {
  cout << "a is greater than b";
}
```

While loop

```
int num = 1;
while (num < 6) {
  cout << "Number: " << num << endl;
  num = num + 1;
}
```

While

```
int a = 0;
do {
  cout << a << endl;
  a++;
} while(a < 5);
```

Do while

For loop

```
int myArr[5];

for(int x=0; x<5; x++) {
  myArr[x] = 42;
}
```

Switch statement to analyze cases

```
int age = 25;
switch (age) {
  case 16:
    cout << "Too young";
    break;
  case 42:
    cout << "Adult";
    break;
  case 70:
    cout << "Senior";
    break;
   default:
    cout << "This is the default case";
}
```

And / Or

```
int age = 20;
if (age > 16 && age < 60) {
  cout << "Accepted!" << endl;
}
```

And: &&

```
int age = 16;
int score = 90;
if (age > 20 || score > 50) {
    cout << "Accepted!" << endl;
}
```

Or: ||

Arrays

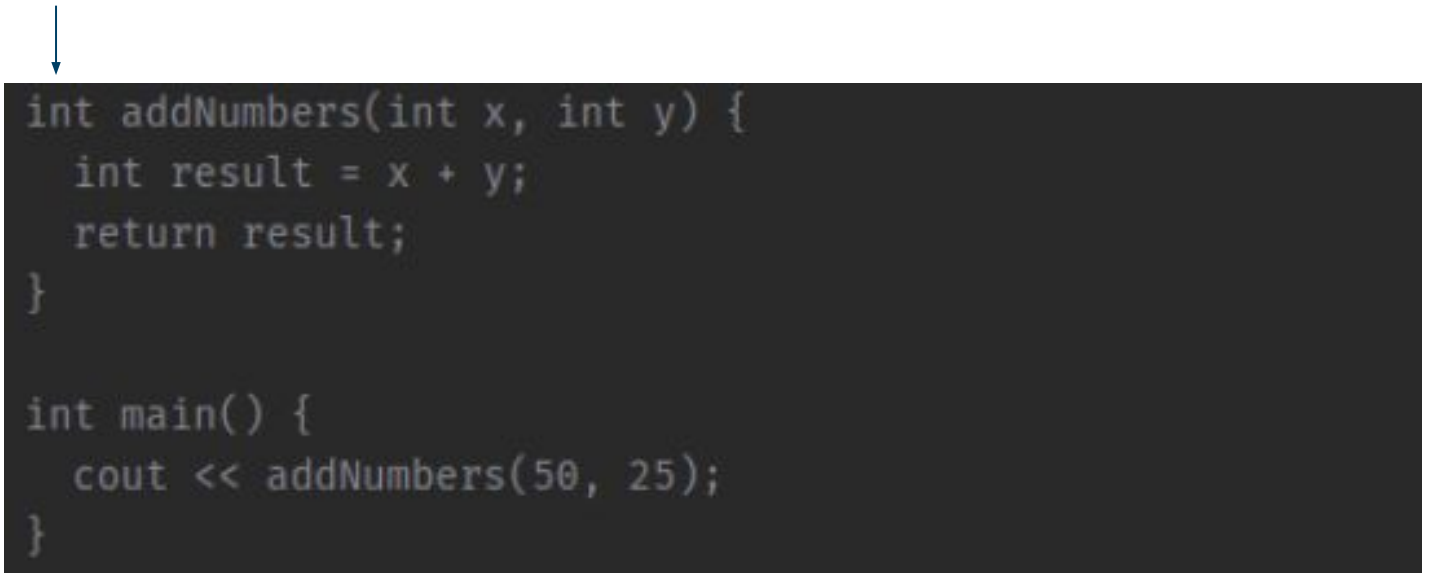
```
int arr[] = {11, 35, 62, 555, 989};
int sum = 0;

for (int x = 0; x < 5; x++) {
  sum += arr[x];
}

cout << sum << endl;
```

Multidimensional arrays

```
int x[2][3] = {
  {2, 3, 4}, // 1st row
  {8, 9, 10} // 2nd row
};
```

Functions

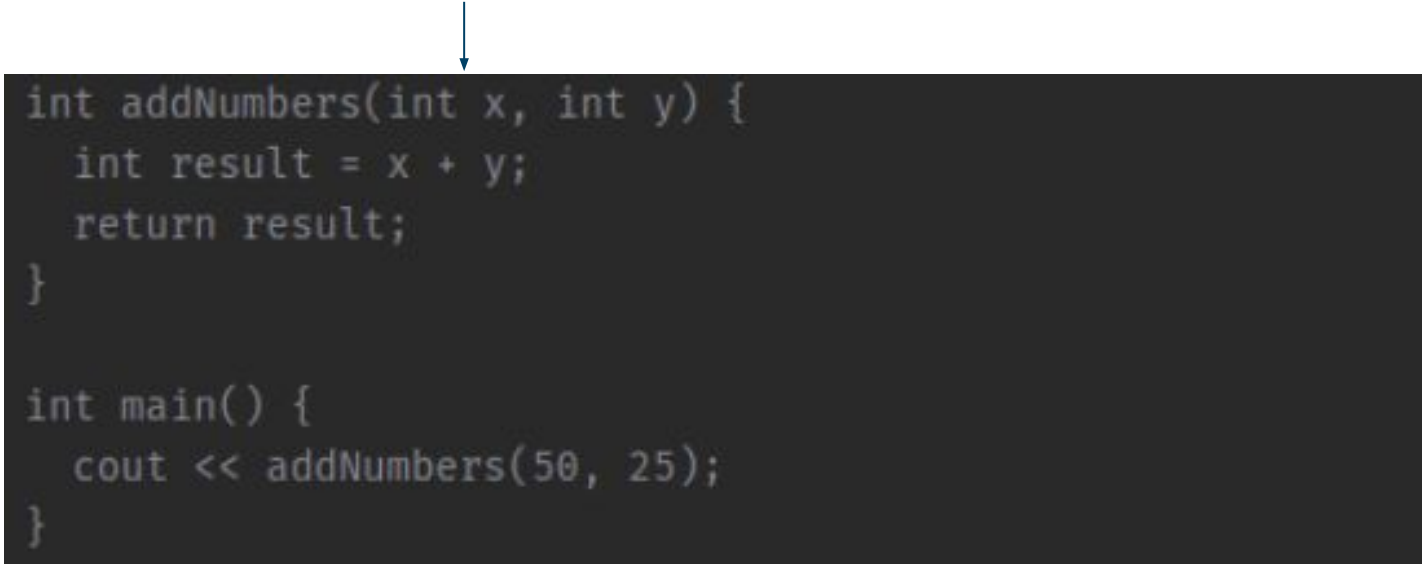
must indicate the return type, if not return use "void"



```
int addNumbers(int x, int y) {
  int result = x + y;
  return result;
}

int main() {
  cout << addNumbers(50, 25);
}
```

Functions

must indicate the type of parameters



```
int addNumbers(int x, int y) {
  int result = x + y;
  return result;
}

int main() {
  cout << addNumbers(50, 25);
}
```

Default parameters

```
int volume(int l=1, int w=1, int h=1) {
  return l*w*h;
}

int main() {
  cout << volume() << endl;
  cout << volume(5) << endl;
  cout << volume(2, 3) << endl;
  cout << volume(3, 7, 6) << endl;
}
```

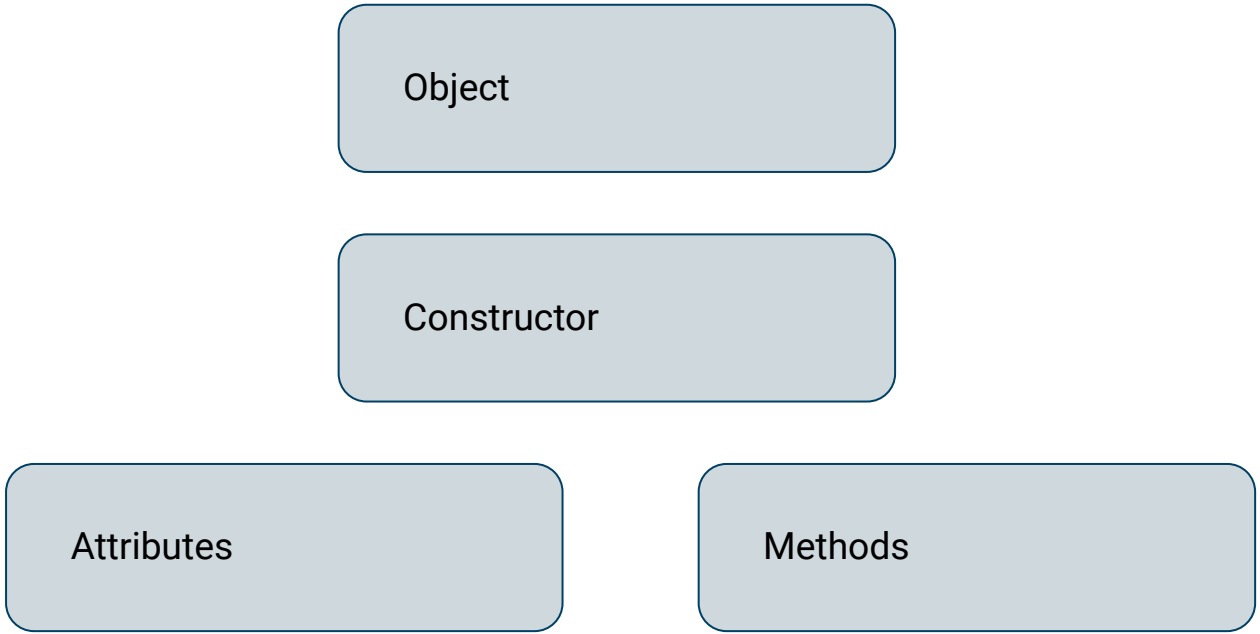
Default parameters can be set on python as well.

OOP paradigm

- C++ al igual que python permite programar orientado a funciones y orientado a objetos.

OOP Paradigm

- General structure



Classes

Attributes


```
class myClass {
  private:
    string name;
  public:
    myClass() {
      cout <<"Hey";
    }
    void setName(string x) {
      name = x;
    }
    string getName() {
      return name;
    }
};

int main() {
  myClass myObj;

  return 0;
}
```

Classes

Constructor

```
class myClass {
  private:
    string name;
  public:
    myClass() {
      cout <<"Hey";
    }
    void setName(string x) {
      name = x;
    }
    string getName() {
      return name;
    }
};

int main() {
  myClass myObj;

  return 0;
}
```

Classes

Functions


```
class myClass {
  private:
    string name;
  public:
    myClass() {
      cout <<"Hey";
    }
    void setName(string x) {
      name = x;
    }
    string getName() {
      return name;
    }
};

int main() {
  myClass myObj;

  return 0;
}
```

Classes

Create the object in the main

```
class myClass {
  private:
    string name;
  public:
    myClass() {
      cout <<"Hey";
    }
    void setName(string x) {
      name = x;
    }
    string getName() {
      return name;
    }
};

int main() {
  myClass myObj;

  return 0;
}
```

Example

You have to create a class, named Student, representing the student's details, as mentioned above, and store the data of a student. Create setter and getter functions for each element; that is, the class should at least have following functions:

- get_age, set_age
- get_first_name, set_first_name
- get_last_name, set_last_name
- get_standard, set_standard

Also, you have to create another method to_string() which returns the string consisting of the above elements, separated by a comma(,). You can refer to stringstream for this.

Problem taken from https://www.hackerrank.com/

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.

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

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

    return 0;
}
```

Your Output

```
0x7ffffa2568bc
```

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;
}
```

Stack vs Heap Memory

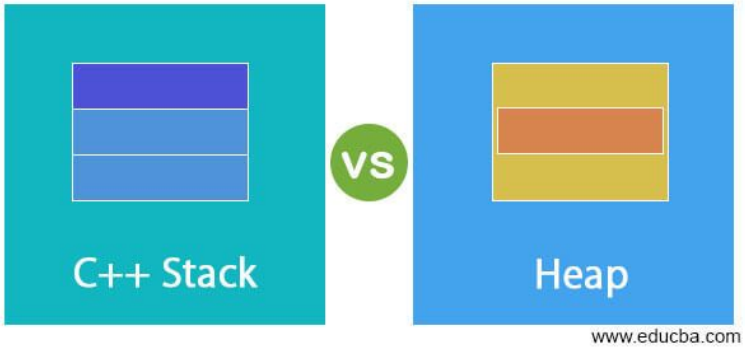
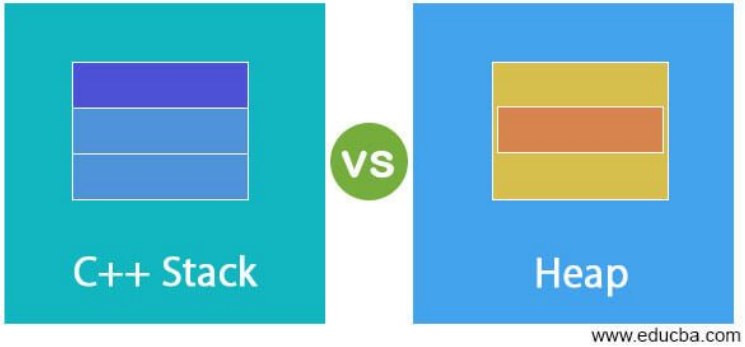


Image taken from https://www.educba.com/c-stack-vs-heap/

Stack vs Heap Memory



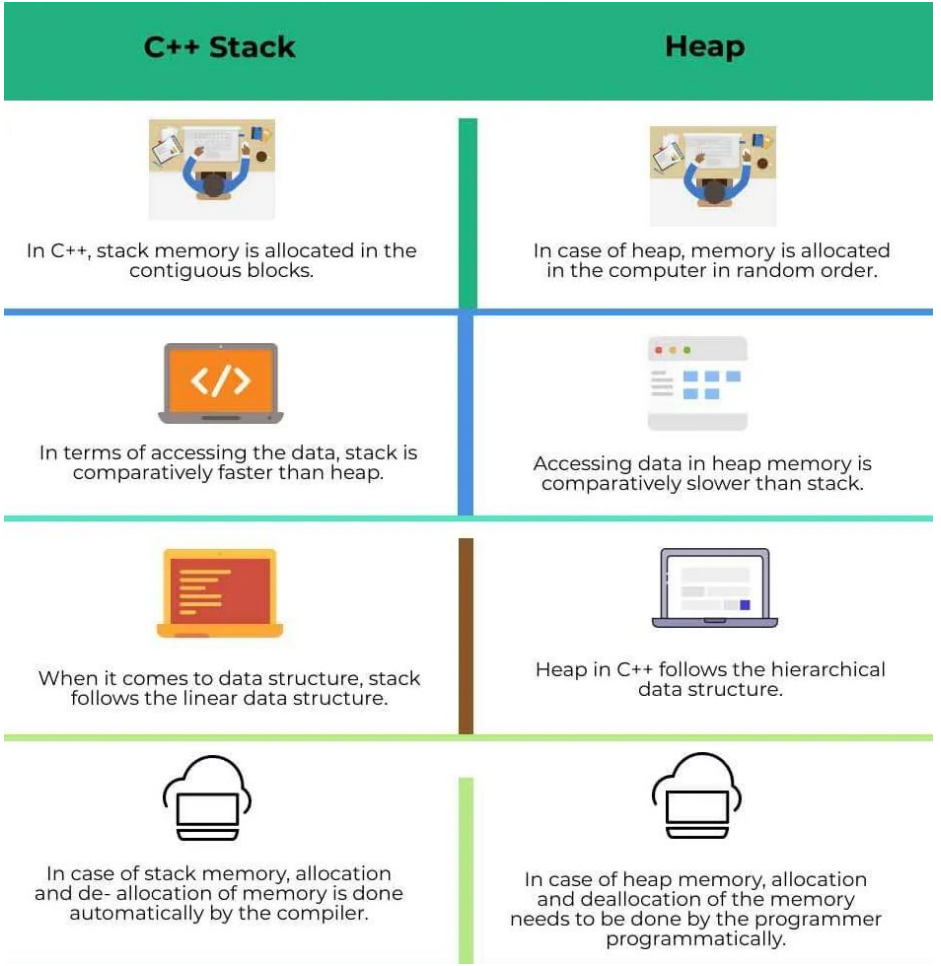
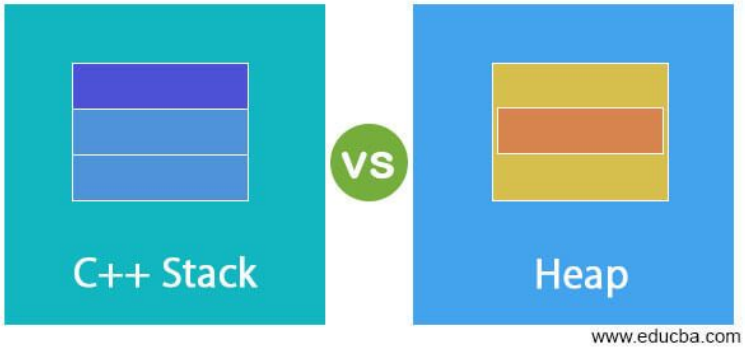


Image taken from https://www.educba.com/c-stack-vs-heap/

Stack vs Heap Memory



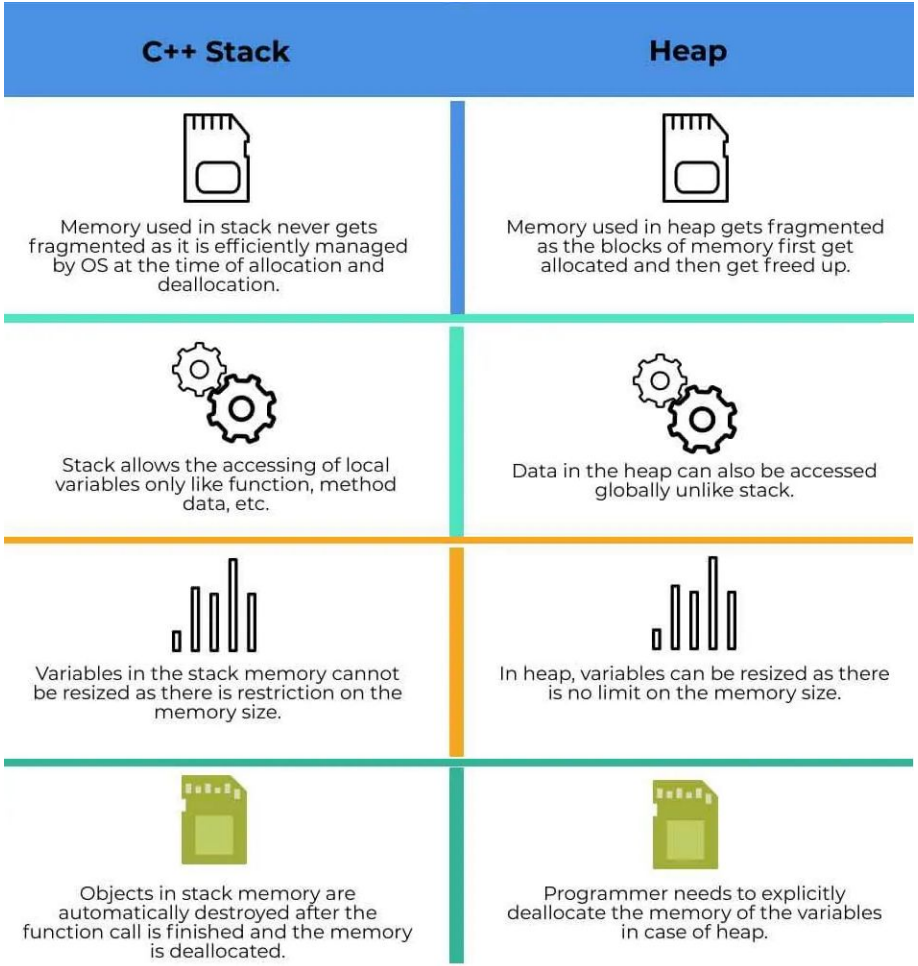


Image taken from https://www.educba.com/c-stack-vs-heap/

Create a variable in heap

```cpp
#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.

Create a variable in heap

```cpp
#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 absoluted 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
```

Problem taken from https://www.hackerrank.com/

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/