Module-1: C++, Vectors, and Pairs

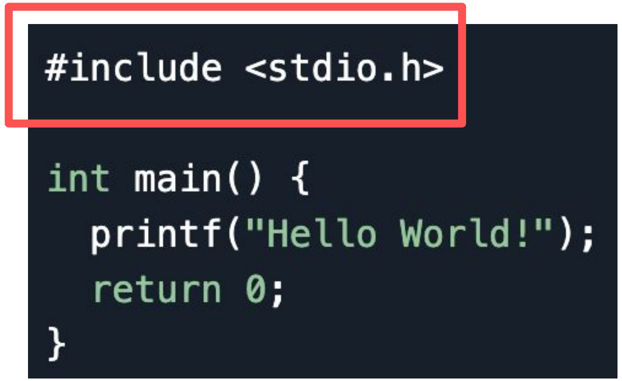
CRUx Bootstrap

1.A: Getting started with C++

C++ is very similar to C, which you have already learnt in CS F111.

There are a few differences that you need to keep in mind, however.

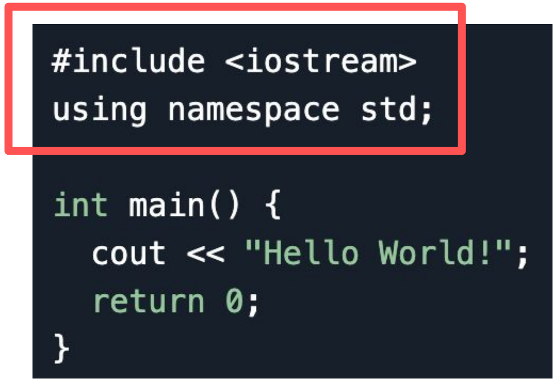
The first difference is in the starting line(s) of code (do not worry about what these lines mean for the time being).



```
#include <stdio.h>

int main() {
  printf("Hello World!");
  return 0;
}
```

Hello World in C

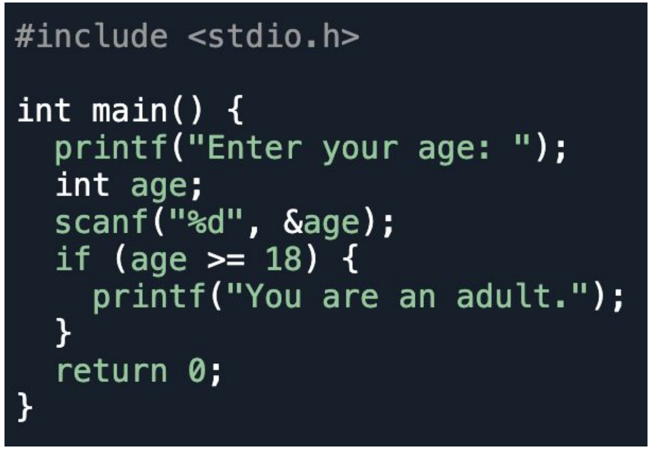


```
#include <iostream>
using namespace std;

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

Hello World in C++

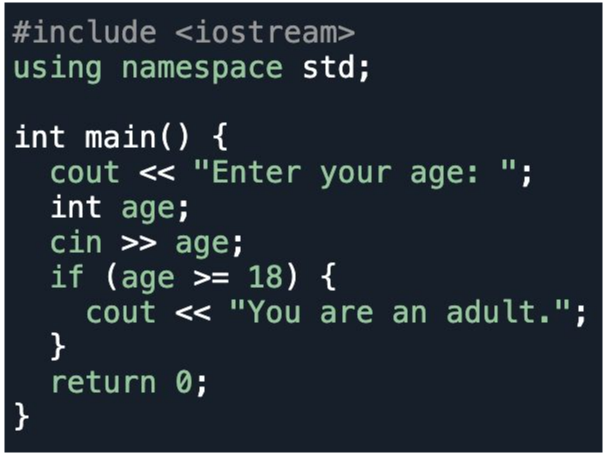
The second difference is the way you take input and output.



```c
#include <stdio.h>

int main() {
  printf("Enter your age: ");
  int age;
  scanf("%d", &age);
  if (age >= 18) {
    printf("You are an adult.");
  }
  return 0;
}
```

I/O in C



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

int main() {
  cout << "Enter your age: ";
  int age;
  cin >> age;
  if (age >= 18) {
    cout << "You are an adult.";
  }
  return 0;
}
```

I/O in C++

For more information, check out the following articles.

1. GFG article on cin
2. GFG article on cout

1.B: Basic problem solving

In competitive coding, most problems at a beginner-level typically require very little topic-knowledge - having basic problem-solving intuition is enough.

This is a primary difference between CC and DSA-like problems, which on the contrary do require having background knowledge.

We have assembled a few problems that require basic problem-solving skills. Can you solve them? Feel free to reach out to your mentors if you are stuck.

Note: do not worry about the efficiency of your program for the time being. We will get to that in a later module.

1.1: You are given two integers - **x** and **y**. Your task is to determine whether **x^y** is strictly greater than **y^x** or not. **x** and **y** are guaranteed to be relatively small, so do not worry about integer overflow. Print **"Yes"** if **x^y** is greater and **"No"** otherwise.

Hint: Pay attention to the phrase "strictly greater". This implies that if both are equal, you should still print **"No"**.

Input: the only line of input consists of two integers **x** and **y**.

Output: print **"Yes"** or **"No"**.

Examples:

3 5

Yes

(3^5 = 243, 5^3 = 125)

2 4

No

(2^4 = 16, 4^2 = 16)

1.2: You are given three integers **a**, **b**, and **c**, denoting the lengths of the sides of a triangle. Do they form a right-angled triangle?

Input: the only line of input consists of three integers **a**, **b**, and **c**.

Output: print **"Yes"** or **"No"**.

Examples:

3 4 5

Yes

(It is possible to build a right-angled triangle by taking 5 as the length of the hypotenuse)

4 5 7

No

5 12 13

Yes

(It is possible to build a right-angled triangle by taking 13 as the length of the hypotenuse)

1.3: Codeforces 4A: Watermelon

1.4: Codeforces 1A: Theatre Square

1.5: Codeforces 977A: Wrong Subtraction

1.6: Codeforces 200B: Drinks

1.C: Vectors

Vectors are similar to arrays, but there is one key difference - they are dynamic, which means that do not necessarily have a fixed size.

To use vectors in your C++ code, you must add “#include <vector>” at the beginning of your program (this imports the vector C++ module).

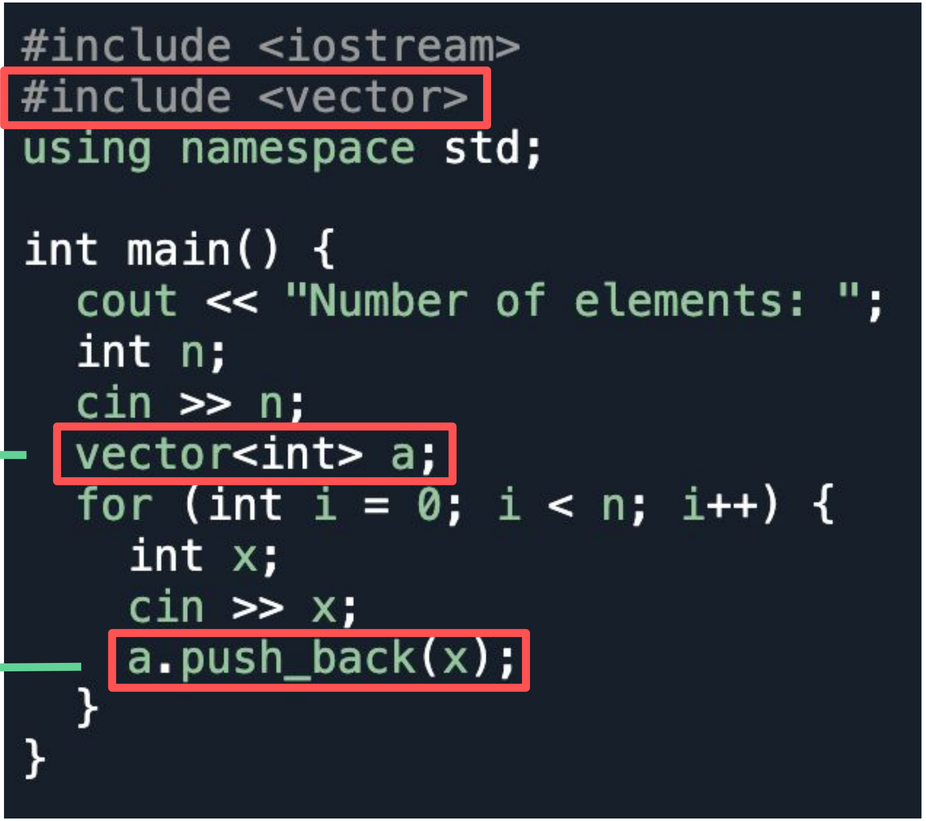
Reading material:

1. GFG article on vectors
2. List of methods that can be performed on a vector

1. Including the vector header file

2. Defining a vector of integers

3. Adding integers to a vector



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

int main() {
  cout << "Number of elements: ";
  int n;
  cin >> n;
  vector<int> a;
  for (int i = 0; i < n; i++) {
    int x;
    cin >> x;
    a.push_back(x);
  }
}
```

Inputting **n** elements into a C++ vector

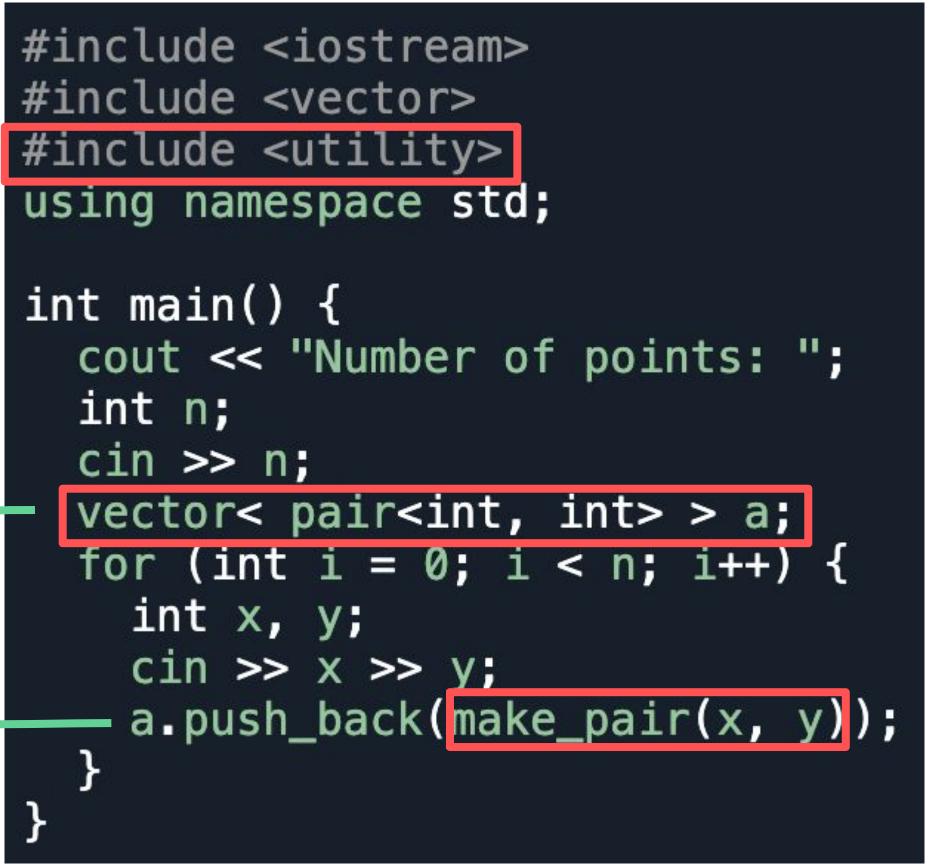
1.D: Pairs

As the name suggests, a pair is a simple object that can hold a pair of items.

To use pairs in your C++ code, you must add “#include <utility>” at the beginning of your program.

Reading material:

1. GFG article on pairs
2. The make_pair() function



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

int main() {
  cout << "Number of points: ";
  int n;
  cin >> n;
  vector< pair<int, int> > a;
  for (int i = 0; i < n; i++) {
    int x, y;
    cin >> x >> y;
    a.push_back(make_pair(x, y));
  }
}
```

1. Including the utility header file

2. Defining a vector of pairs

3. Making a pair of integers

Pairs can be combined with vectors to do powerful things

1.7: You are given a list of integers. Find the number of inner elements (all elements except the first and last) that are smaller than both its neighbouring elements.

Input:

The first line consists of an integer **n** - the size of the list.

the second line consists of **n** integers.

Output: print a single integer.

Examples:

5

1 4 2 3 5

1

7

2 1 2 1 2 1 2

3

1.8: You are given a list of integers. Find the smallest and largest elements,

Input:

The first line consists of an integer **n** - the size of the list.

the second line consists of **n** integers.

Output: print two integers - the smallest and largest elements.

Examples:

8

3 4 3 8 7 2 3 5

2 8

3

2 3 1

1 3

1.9: Codeforces 158A: Next Round

1.10: Codeforces 116A: Tram

1.11: Codeforces 268A: Games

1.12: Codeforces 231A: Team