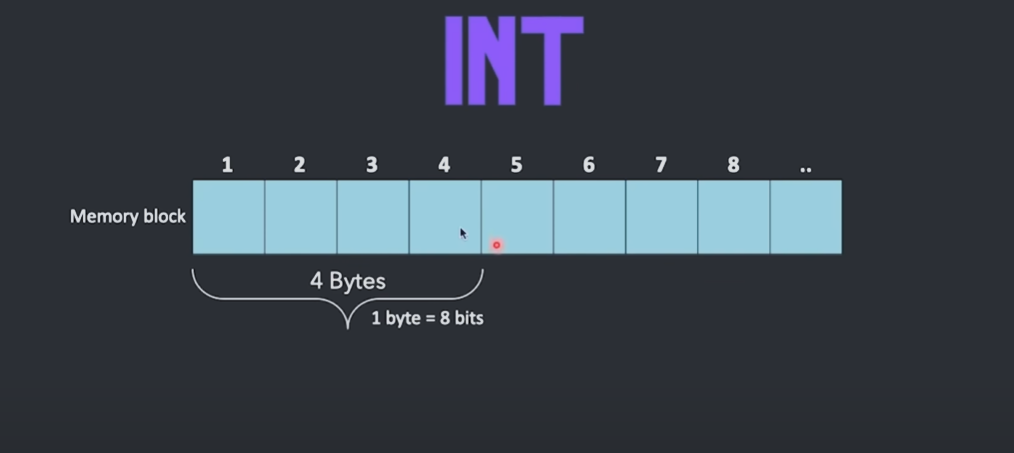
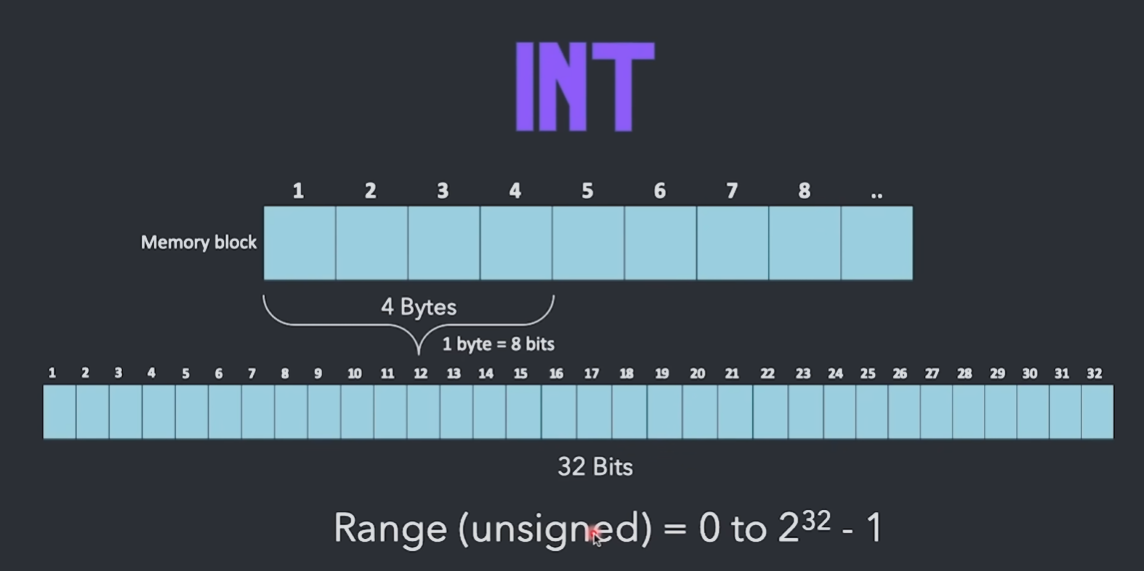
**Int occupies 4 bytes**



1 byte = 8 bits

* TO write negative integer we use msb which is the most significant bit or 1st bit as shown in the second figure.
* 1 - -ve number
* 0 = +ve number
* So in signed integer we can write up to 2^31 to 2^31-1.
* Whereas in unsigned, we can go upto 0 to 2^32 – 1.

use of 0 is only one cuz +ve and -ve zero don’t exist and the one extra size will go to -ve side so it can store up to 2^31 bytes.

**FLOAT – 4 byte**

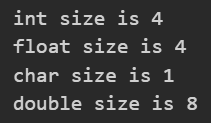
**Double – 8 byte**

**Character – 1 byte**

**Bool – 1 byte (true or false)**

**Simple program printing size**

**#include<iostream>**

**using namespace std;**

**int main(){**

**int a;**

**float b;**

**char c;**

**double d;**

**cout<<"int size is "<<sizeof(a)<<endl;**

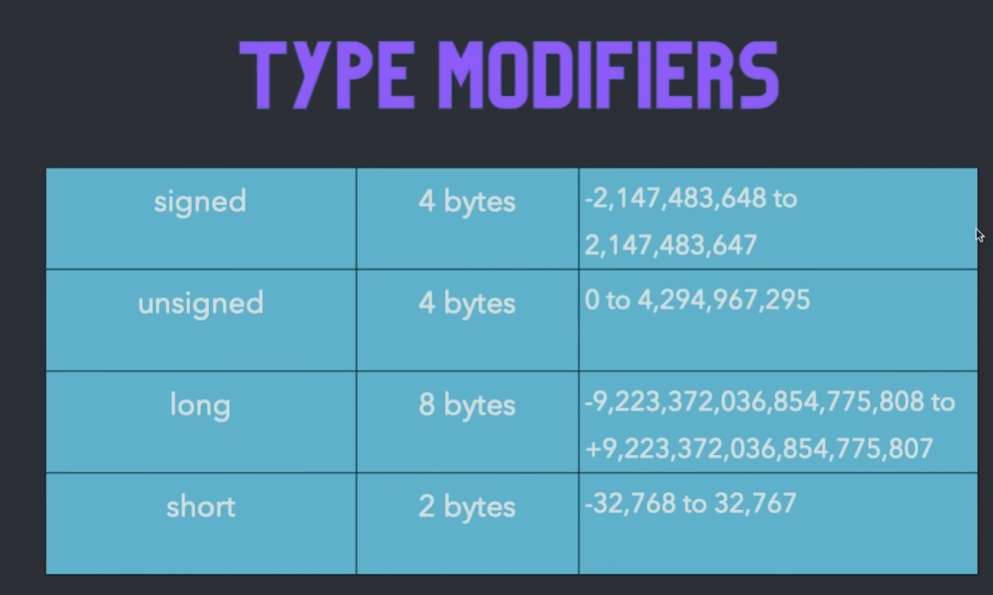
**cout<<"float size is "<<sizeof(b)<<endl;**

**cout<<"char size is "<<sizeof(c)<<endl;**

**cout<<"double size is "<<sizeof(d)<<endl;**

**return 0;**

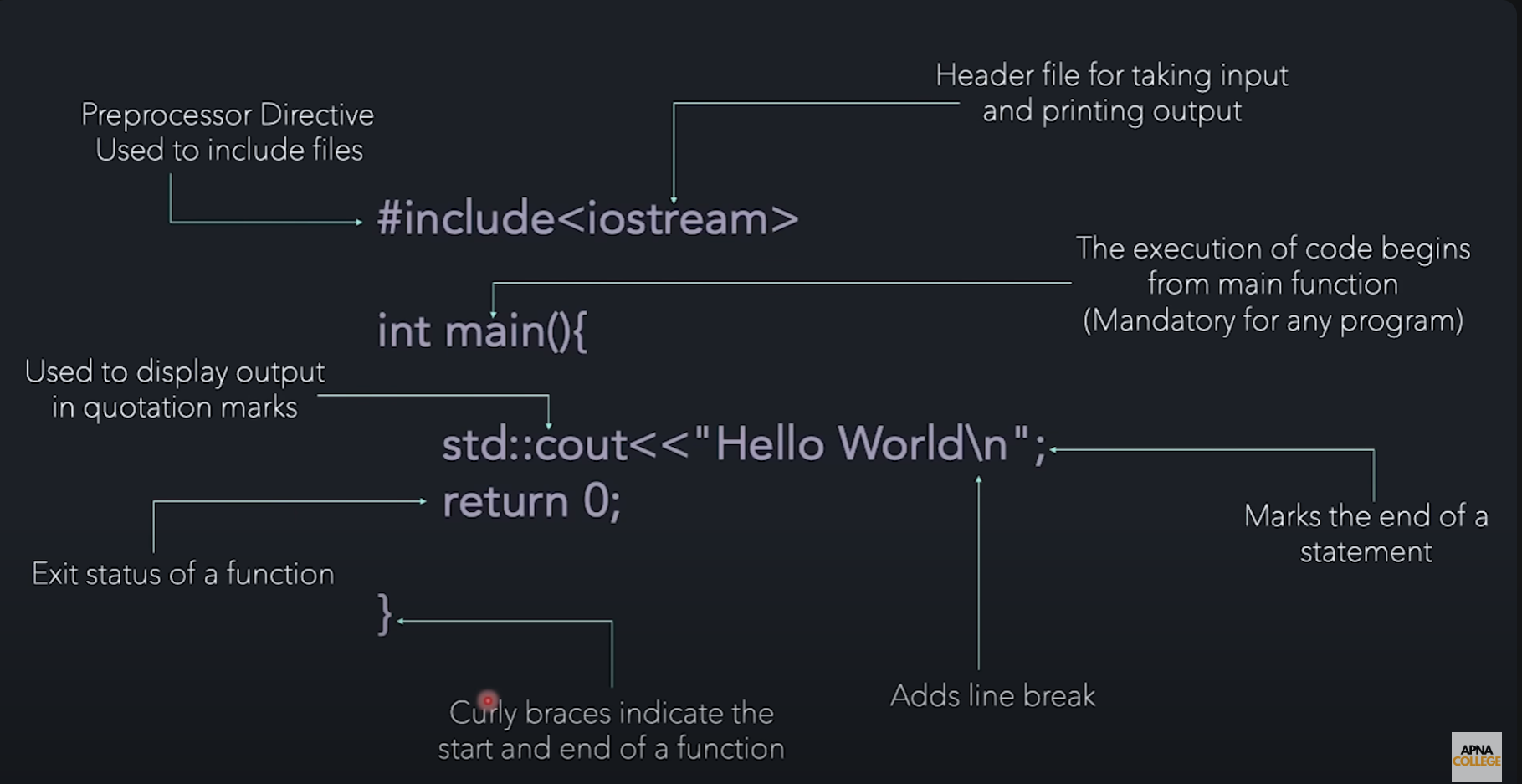
**}**

****

Type modifiers are used in order to increase the default size of data types



Preprocessor Directive

#include<iostream> - for i/p and o/p (same as stdio.h in c) 

Namespace

A namespace is a C++ language feature introduced to organize code into logical groups and prevent naming conflicts. It provides a scope for the identifiers declared within it, allowing you to define entities with the same name in different namespaces without causing conflicts.

namespace my\_namespace {

// Declarations go here //way of defining namespace

int my\_variable;

void my\_function();

class MyClass;

}

While writing program for hello world I found that we use std::cout<< helloworld is used so here **std is a namespace and cout falls under that namespace.**

In bigger programs, we can just import the std namespace outside our main function so we don’t have to write it every time.

For eg:

#include <iostream>

using namespace std;

int main(){

    cout<<"HEllo WOrld";

    return 0;

}

**So this was a lil bit of convo with chat gpt about the function of “<<”.**

This is called insertion operator and it can also be used as a format specifier in c like %d and all to put in values in output stream.

We can achieve same like shown in the convo below.

Yes, that's correct. In C, when using **printf** for formatted output, you typically use format specifiers like **%d**, **%s**, etc., to indicate where and how variables should be inserted into the output string. For example:

cCopy code

int x = 42; printf("The value of x is: %d\n", x);

Here, **%d** is a format specifier for an integer variable (**x** in this case). The value of **x** will be inserted into the output string at the position of **%d**.

In C++, when using **cout**, you don't use format specifiers like **%d**. Instead, you can directly insert variables into the output stream using the **<<** operator. For example:

cppCopy code

int x = 42; std::cout << "The value of x is: " << x << std::endl;

Here, **<<** is used to insert the value of **x** directly into the output stream. The **<< x** part of the expression means "insert the value of **x** into the output stream at this point".

So, in C++, you use the **<<** operator to separate output strings and values, while in C, you use format specifiers within the format string to indicate where and how variables should be inserted.

Input and Output

#include <iostream>

using namespace std;

int main(){

    cout<<"HEllo WOrld";

    int a;

    cout<<"write a number";

    // std::cin.ignore(); //used to ignore buffer

    cin>>a;

    cout<<"you entered "<<a;

    return 0;

}

**Learnings:**

1. We use << for c out and >> for cin
2. Cin is like scanf for c but here format shapers are not used.
3. You can display text like cin>>"enter a num">>a; like you do it in cout.(its only for taking input)
4. 2 different operators are used while taking input and displaying output as discussed in no 1.
5. Conversation about operators:

In C++, the **>>** operator is called the extraction operator when used with input streams like **cin**. It's used to extract data from the input stream and store it into variables.

Here's how it works:

1. **Extraction**: The **>>** operator takes data from the input stream (e.g., keyboard input) and stores it into the variables on its right-hand side.
2. **Chaining**: Like the insertion operator **<<** for output streams, the extraction operator **>>** can also be chained together to extract multiple values from the input stream sequentially.

For example, in the code:

cppCopy code

int a; cin >> a;

The **>>** operator reads data from the input stream (e.g., keyboard input) and stores it into the variable **a**.

So, **>>** is called the extraction operator because it extracts (or reads) data from the input stream. It's primarily used in conjunction with input operations, such as reading user input from the console using **cin**.

also tell about <<

Certainly! In C++, the **<<** operator is called the insertion operator when used with output streams like **cout**. It's used to insert data into the output stream for displaying.

Here's how it works:

1. **Insertion**: The **<<** operator takes data (such as variables, strings, etc.) and inserts it into the output stream (e.g., console output).
2. **Chaining**: The insertion operator **<<** can be chained together to insert multiple values into the output stream 0073equentially.

For example, in the code:

cppCopy code

int a = 42; cout << "The value of a is: " << a << endl;

The **<<** operator inserts the string **"The value of a is: "** followed by the value of the variable **a** into the output stream represented by **cout**.

So, **<<** is called the insertion operator because it inserts (or writes) data into the output stream. It's primarily used in conjunction with output operations, such as printing data to the console using **cout**.

Top of Form

**Access Sepcifier**

**public:  
private:**

**protected:**