if (a>b) & (a>c), print a
else if (b>a) & (b>c), print b
else print c

28/01/2023

Programming language
Language in which we are giving instructions to our computer. Also note that every programming language will have its own compiler / interpreter
Ex → C++, C, Java, Python are programming language

Compilation process
We will be writing the code in the high level language but computer won't be able to understand it & we need a translator & here compiler will act as a translator. Compiler will convert the high level language to machine understandable language. Machine language means binary language i.e 0s & 1s.

Compiler A compiler takes a program as whole and generates intermediate machine codes. This translates the entire source code in a single run.
Interpreter The interpreter will translate the program line by line & also it never generates any intermediate machine codes.
IDE The full form is Integrated Development Environment. Tool on which we can write the code. It provides the environment, also provides various features & hence it increases our productivity.

Scanned with CamScanner

Ex → Codeblocks, VS code, Sublime, Xcode

Writing our first program

1) int main() { } ← Scope of int main()

This is similar to start component we have done in the flowchart. Compiler will always search for this & from here the execution of the program will start.

2) cout << "Namaste Bharat";

This is used to print on the console/screen but it gives an error that is the use of undeclared identifier. This means that compiler does not know what is cout. Before we use cout, we need to import its code so that we can use this functionality.

3) #include <iostream> ← File having implementation of cout

This has the code of implementation of cout but again it gives error that is, do you mean std :: cout ?

We use std :: cout << "Hello";
↳ This is a namespace

Namespace

This is basically some area and implementation of cout will be different in different namespaces. But we have to use the standard implementation & hence we wrote std :: cout
↳ standard namespace

assume {

Scanned with CamScanner

Namespace is a region where the scope of identifiers have been defined. For example → scope of cout is defined in std namespace

But this is a cumbersome task of writing std::cout again & again & hence we can define/declare which namespace to use.

using namespace std; → Now we can use functionality of print simply by cout <<

cout << "Hi" ; → Terminator

↓

For printing → Part of syntax & these symbols are used to print on standard output display

"Hi" → Hi is a text (string) written inside double inverted commas

Note→
```
cout << 2;
cout << "2";
cout << '2';
```
These will be printing 2 on the screen

Also note that << is known as insertion operator.
Single inverted commas means character.
Double inverted commas means string.

Next line
To go to the next line, we use keyword endl.

Scanned with CamScanner

```
cout << "Hi" ;
cout << "Hello" ;
```
} HiHello

```
cout << "Hi" << endl ;
Cout << "Hello" ;
```
} Hi
Hello

Note→ There is one more way to add new line i.e new line character \n.

```
cout << "Hi \n" ;
cout << "Hello" ;
```
} Hi
Hello

Logically there is no difference b/w endl & \n.

Taking input from user
For printing we use cout, but for taking input from the user, cin is used.
But for taking input, we have to store it in memory & that memory space is given a name

```
int a ;
cin >> a ;
```
→ Taking integer memory space & give it name a.
↳ To take input & send it to memory space a.

Note→ To complete giving input, we need to hit enter key.

Comments in C++
These lines are ignored by compiler & hence

Scanned with Cam

are not executed.

→ We have to add this to make line comment.

// This is a comment

Comments can be used to make notes. Also there is a shortcut to make a line comment is ctrl + /

Variable

Variables can be defined as named memory location so that we can refer it in future.

int a = 7; ⟶ [7] (a) cout << a ; → 7

Data type

This tells what type of data is stored.

Char, short, int, float, double are the data types.

int a = 7;
- int → data type
- a → variable
- 7 → value assigned to a

There can be various types of data types

(i) Built-in ⇒ Already defined. Ex→ int, float, void etc.
- float → for decimals
- void → means empty & no data value

Note → i/p ⟶ [function] ⟶ o/p

int main () {
- int → return type

void main () {
- void → means return nothing

3

Scanned with Cam

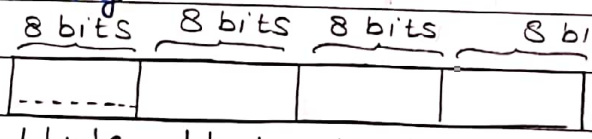
Bytes (Also depends on machine)

	Size	Range
char	1	-128-127
Short	2	-32768-32767
int	4	-21..8 - 214..7
long	4	-21.. 8-214..7
long long	8	-92...8 - 92..7
float	4	3.4 e +/- 38
double	8	1.7 e +/- 308

32 bit CPU → can be different for 64 bit CPU

int datatype

Takes up 4 byte of memory i.e 4x8 = 32 bits of memory.



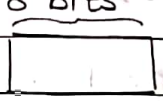
As each bit can have either 0 or 1 value, then there are 2 combination possible for each bit & hence 2^{32} are possible.

Note

char datatype

```
char ch = 'a';
```

↳ 1 byte



2^8 combination are possible as each bit have either 0 or 1 & hence total 8 bits.

$2 \times 2 \text{ -- } \times 2 = 2^8$ (8 times)

**

Scanned with Cam

Now question comes that whether character a is stored or not. But a number is stored instead & all the characters are mapped to ASCII value i.e a numeric value & this numeric value is converted into 0s & 1s & then stored.

a → 97 → converted to 0s & 1s and then stored in memory

boolean data type
Can have only 2 value i.e either true or false.

bool flag = true; → data type
cout << flag; → gives o/p 1.

bool flag = false;
cout << flag; → gives o/p as 0.

Note → true → 1 } Similar
false → 0 }

** The space taken by boolean is 1 byte but it is sufficient to represent it in 1 bit but still 1 byte is taken because smallest / minimum memory that can be used is 1 byte. There is memory wastage also.

Smallest addressable unit is 1 byte.

float & double data type
These are used for representing decimal numbers.

Scanned with CamScanner

```
float f = 1.2;
double f = 1.2;
```

But which one to use? Double is more precise than float & also float takes up less memory space. The one taking more space can represent much bigger number.

Ques→ As we know that 256 characters can only be stored in char (0 to 255) but what if we do
char ch = 256; ?
This is basically the overflow condition & error comes i.e overflow in conversion from int to char changes value from '256' to "\000". Also we can say last 8 bits will be copied.

(ii) Derived data types
Primitive data type are used to make these. Ex→ Arrays, pointer etc.

(iii) User defined data type
Make our own data type. Ex→ structures, union, classes & enumerations.

Variable naming conventions
1) Can contain letters, digit & underscores
2) Names must begin with letter or underscore.
3) Names are case sensitive
4) Names can not have spaces & no special character other than underscore can be used.

sizeof ()
This returns the size of data type in bytes

Scanned with CamScanner

sizeof (int) ; → 4 } in bytes
sizeof (char) ; → 1 }

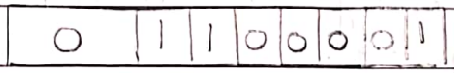
Storage of data

(i) Positive numbers

int a = 5;

32 bits

char ch = 'a'; → Just convert the ASCII value to binary & then it will be similar to that of integer.



8 bits

It is important to note that first bit from left will be 0 if +ve number is there. It will be 1 if -ve number is there.

(ii) Negative number

Will be stored in form of 2s compliment.

Way to find 2's complement

↳ 1s complement + 1

Flip 0 to 1 & 1 to 0

7 → 00000111 } Just for understanding 8 bits.

11111000 → 1s complement

```
  11111000
         1
 ---------
  11111001 → 2s complement
```

Note → In binary 1+1 = 10.

Scanned with CamScanner

int a = -5

Step1 Ignore negative sign.

5

Step2 Find binary equivalent of 5

000---- 0101

Step3 Find 2s complement

1111--1010

1

-ve no. ← ①1 1-- 1011 → 2s complement

This way -ve number is stored in memory.

How to read from memory?
Just find 2s complement

000--0100

1

000--0101 → 5

But we have already seen that first bit in memory representation was 1 & hence it is a negative number

Ans → -5

Interesting problem

22	31	43	7

→ each of 1 byte

How we get to know whether we have to read all the 4 bytes or just 1 byte.

Scanned with Cam

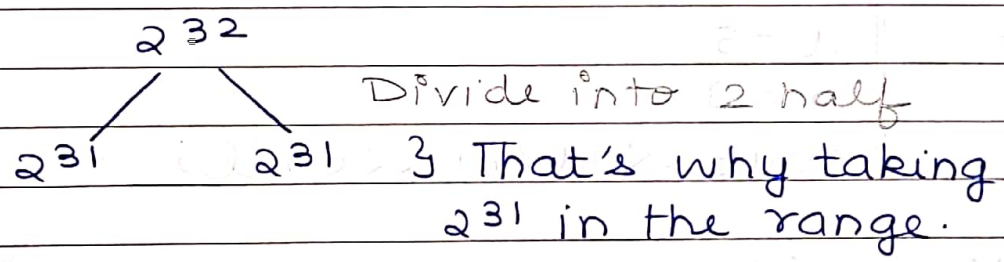
This is known by the data type used.

Signed vs unsigned data type
In signed we can store all the numbers whether +ve, -ve or 0.
In unsigned we can not store the negative values.

For example → unsigned int will have range of 0 to $2^{32}-1$

Note → By default, data type will be signed.

Range of signed int will be -2^{31} to $2^{31}-1$.



Generic for n bits
Signed → -2^{n-1} to $2^{n-1}-1$
Unsigned → 0 to $2^{n}-1$

Typecasting
Converting one type of data type to another data type is known as typecasting.

char ch = 97; → Integer to character
cout<<ch; → gives a as output

int num = 'b'; → Character to integer
cout << num; → gives 98 as output

Scanned with CamScanner

Implicit Type conversion
This is automatic typecasting. The above examples were of implicit type conversion.

Explicit Type conversion
This is done manually. This uses ( ) operator for typecasting
float a = (float) 2 ;
↳ here data type will be mentioned.

Operators in C++

1) Arithmetic operators
+, −, *, /, % are arithmetic operator.

a = 5
b = 3
cout << (a / b) << endl ; → This gives o/p as 1.

Note → $\frac{\text{int}}{\text{int}}$ = int , $\frac{\text{float}}{\text{int}}$ = float , $\frac{\text{double}}{\text{int}}$ = double
Bigger data type will always dominate.

$\frac{\text{int}}{\text{float}}$ = float , $\frac{\text{float}}{\text{float}}$ = float

2) Relational operators
>, <, >=, <=, != , ==
!= → not equal to
== → equal to
This will give the boolean output.

Scanned with CamScanner

3) Assignment operator
= int a = 3 ; → 3 has been assigned to a

4) Logical operator
Used when there are multiple conditions.
&& → Logical AND (AND gate)
|| → Logical OR (OR gate)
! → Logical NOT (NOT gate)

(i) AND
0 && 0 → 0 0 && 1 → 0
1 && 0 → 0 1 && 1 → 1

(ii) OR
0 || 0 = 0 1 || 0 = 1
0 || 1 = 1 1 || 1 = 1

(iii) NOT
! 0 = 1 ! 1 = 0

Note → (cond1) && cond2 && cond3
↳ If it becomes false, then no need to check other conditions.

Scanned with CamScanner