

DATA STRUCTURE & ALGO

1.0 Binary (0,1) → Language of computer
 $(0,1 \rightarrow \text{Low/High Voltage})$

Ques We have two hourglass of 7hr & 4hr.
 How to measure 9mins?

In 4 hours, the hourglass II will be empty, further we will flip the hourglass II for 3 hr till hourglass I is empty. Now 7 hours have elapsed and we have 1 hr sand in hourglass II, let the sand flow & flip hourglass I. After eighth hour, hourglass II will be empty, again we will flip hourglass I & II. When hourglass I will be empty, 9 hours will be elapsed.

1.1 PROGRAM: Set of instructions that tell a computer what to do.

1.2 Compiler: Computer program that translates HLL to Machine Language. (Intermediate Code)

1.3 Interpreter: Executes the instructions directly & removes the process of converting to intermediate code

1.4 C++ → Cross-platform language, Developed by Bjarne Stroustrup.

Ques 3 light bulbs in a room, 3 button outside, can go inside once, how to find the light & its switch

First, turn on a light for 30 mins, turn it off & enter the room,

↳ WARM LIGHT → First Switch

↳ Glowing Light → Second Switch

↳ OFF LIGHT → Third Switch

Ques 3ltr jug + 5ltr jug. How to measure 4ltr

↳ Fill 3ltr jug (0, 3)

↳ Pour 3ltr to 5ltr & fill 3ltr again (3, 3)

↳ Pour 3ltr to 5ltr again (5, 1)

↳ Empty 5ltr & pour 3ltr to 5ltr (1, 0)

↳ Fill 3ltr (1, 3)

↳ Pour 3ltr to 5ltr (4, 0)

2.0

VARIABLES: Container used to hold data.

Each var should have a unique name

Data Type	Meaning	Size
int	Integer	4
float	Floating-point	4 → Upto 7 Decimal
double	Double floating pt.	8 → Upto 15 Decimal
char	Character	1 → Inside single quote
wchar_t	Wide Character	2
bool	Boolean	1 → false / true
void	Empty	0

2.1 DATA TYPE Modifiers

- ↳ Used to modify the fundamental Data Type.
- ↳ signed, unsigned, long, short

iostream: Input/Output stream in C++ used for input as well as output of a particular value.

- ↳ cout → Standard Output
- ↳ cin → Standard Input
- ↳ cerr → Standard Error

3.0 * Comment: Displays important info about prog.
Compiler skips the comment line

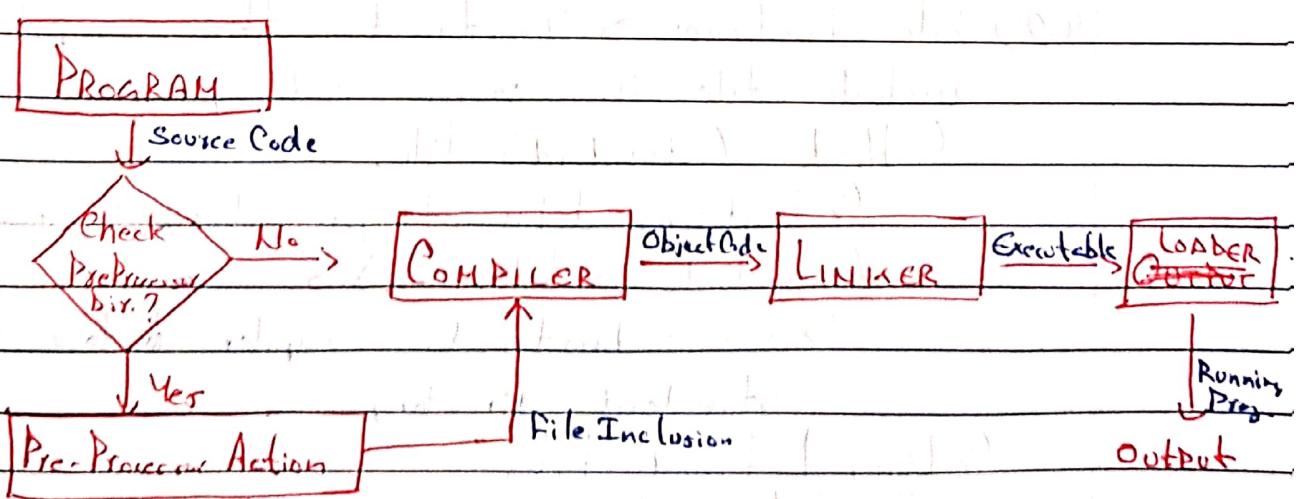
// Single line /* Multi-line Comment */

When & Why Comments?

↳ Make our code more readable

↳ Provide description to algorithm

4.0 * Pre-Processor: Programs that process source code before compilation.



↳ Pre-processor begins with `#` symbol.

4.1) Macro → Piece of code which is given some name.

↳ Whenever the name is encountered, compiler replaces name with code.

`#include <iostream>`

`using namespace std;`

`#define LIMIT 5`] → The limit var defined
int main() { ... }] will replace whenever
`cout << LIMIT << endl;`] called.
`return 0;`

↳ Note: There is no semicolon (`;`) at the end of macro definition

↳ We can also pass arguments to macro & make it as function.

`#define AREA(l, b) (l*b)`] → Here, the macro

`AREA(5, 2)`]

is sep by space b/w arguments &

return value, act like function.

4.2) File Inclusion → Instructs the compiler to include file in source code program.

↳ Header (Standard) files

↳ Contains defn of pre-defined functions

`#include <filename>`]

↳ `<>` → Instructs compiler to look files in standard directory.

↳ User Defined Files

`#include "filename"`

↳ `" "` → Instruct compiler to look in current directory.

4.3) Header File Inclusion

↳ Used in large programs where a single user-defined header is in use by multiple files.

#ifdef MACRO

Used in case of Macro

#endif

#ifndef -- HEADER--

#define -- HEADER--

HEADER FILE CONTENT

#endif

→ Used in header

files so that

linker doesn't link

the content multiple

times.

(github.com/kshitizsaini13/L2-Emulator)

↳ Here, if the macro is defined, the compiler will skip block of code else will simply execute it.

4.4) #undef → Used to undefine a macro.

#undef MACRO.

5.0 Why not use "using namespace std"

↳ It is suggested to use scope resolution operator (::) to import specific functions rather than importing entire std namespace.

~~# using std::cout;~~

↳ std namespace is huge & has hundreds of defn so user may ~~redefine~~ redefine & result in compilation error.

6.0 main()

↳ main() → Entry point of program.

↳ Every C++ program must have a main() function.

7.0

BASIC INPUT / OUTPUT

- ↳ C++ Input / Output are known as streams
- ↳ Input Stream: Keyboard → Main Memory
- ↳ Output Stream: Main Memory → Screen

iostream : (Standard input / output) Contains defn of cin, cout, cerr, etc.

iomanip : (Input / output manipulators) Contains methods for manipulating streams.

fstream : (File Stream) Handler data from file as input or output

↳ cout : Produce o/p on the stream (screen) using the insertion operator (<<)

↳ cin : Read i/p from the keyboard using the extraction operator (<>)

↳ cerr : Standard error stream used to o/p error message immediately. & don't store for later.

↳ clog : Error is first inserted to buffer and then displayed.

8.0

bits/stdc++.h

↳ Using <bits/stdc++.h>

↳ Not available in all compilers

↳ Reduces time of writing include

↳ Increases compilation time.

3.0 NAME SPACE

↳ Allow us to group named entities & remove them from global namespace to reduced namespace.

namespace NAMESPACE { }

}

4.0 endl v/s \n

↳ endl → Inserts a new line & flushes the stream.

↳ \n → Only inserts a new line.

`cout << endl;` = `cout << '\n' << flush;`

\n is performance-wise better unless flushing of stream is reqd.

5.0 PreIncrement v/s postIncrement

↳ preIncrement can be used as left value but postIncrement can never be used as a left value.

$$t+a = 20 \quad \checkmark$$

$$a++ = 20 \rightarrow \text{ERROR}$$

↳ ++a returns a reference to variable to which we can further assign whereas a++ holds the value.

$$\text{int } b = ++a \rightarrow \checkmark$$

$$\text{int } b = a++ \rightarrow \text{ERROR}$$

$$a++ = 20 \quad \text{Illegal}$$

$$\text{G} \boxed{10 = 20}$$

$$a = a + 1$$

7

12.0 PRECISION FOR NUMBERS

(\hookrightarrow) 12.1) `floor()`

(\hookrightarrow) Rounds off to closest integer less than given value.

(\hookrightarrow) 12.2) `ceil()`

(\hookrightarrow) Rounds off to closest integer more than given value.

(\hookrightarrow) 12.3) `trunc()`

(\hookrightarrow) Removes digits after decimal points.

(\hookrightarrow) 12.4) `round()`

(\hookrightarrow) Rounds number to closest integers.

(\hookrightarrow) 12.5) `setprecision()`

(\hookrightarrow) Provides precision to floats when used with fixed keyword.

```
int x = 1.13113
cout << floor(x); → 1
cout << ceil(x); → 2
cout << trunc(x); → 1
cout << round(x); → 1
cout << fixed << setprecision(2) << x; → 1.13
```

13.0 if - elif - else

(\hookrightarrow) Used to specify the code to be executed according to some conditions.

`if (Condition) {`

\hookrightarrow `else if (Condition) {`

\hookrightarrow `else {`

\hookrightarrow `}`

Que

Check if even odd number.

#include <iostream>

using namespace std;

int main() {

int n;

cin >> n;

if (n % 2 == 0) {

cout << "Even";

}

else {

cout << "Odd";

}

return 0;

}

Condition for checking
if a number is
even or odd by
dividing it by 2 and
checking remainder.

Que

Max-Min from two numbers.

void max_min (int n1, int n2) {

if (n1 > n2) {

cout << "Max" << n1;

cout << "Min" << n2;

}

else (n2 > n1) {

cout << "Max" << n2;

cout << "Min" << n1;

}

y

Que

Max from 3 numbers

int max_three (int n1, int n2, int n3) {

if (n1 > n2) {

if (n1 > n3) {

```

cout << return n1;
}
else {
    cout << return n3;
}
}
else if (n2 > n3) {
    cout << return n1;
}
else {
    cout << return n3;
}
}

```

Que Check the type of triangle.

```

if (s1 == s2 && s2 == s3) {
    cout << "Equilateral" << endl;
}
else if (s1 == s2 || s2 == s3 || s3 == s1) {
    cout << "Isosceles" << endl;
}
else {
    cout << "Scalene" << endl;
}

```

14.0 Loops

↳ Loops are used for executing a block of statement repeatedly until a condition is satisfied.

- ↳ ENTRY CONTROLLED : for, while
- ↳ EXIT CONTROLLED : do-while.

14.1) for Loop

for (INITIALIZATION; CONDITION; MODIFICATION)

{
 // STATEMENTS
}

14.2) while Loop

while (CONDITION)
{

 // STATEMENT
}

14.3) do-while loop

do
{

 // STATEMENT
}

} while (CONDITION);

15.0 Jumps (Loops) (Control Flow)

- ↳ Jumps are used to control the flow of loop.
- ↳ break
 - ↳ continue

15.1) continue

↳ used to skip to next iteration of loop.

15.2) break

↳ used to terminate the current loop.

Que

Calculate sum of first n natural.

```
for (int i=1; i<=n; i++) {
```

```
    sum += i;
```

}

```
cout << sum;
```

Here the sum must be initialized to 0 and n must be declared first.

Que

Print table of a number

```
for (int i=0; i<=10; i++) {
```

```
    cout << "n << " * " << i << " = " << n*i;
```

}

Que

Add numbers till +ve are encountered.

```
int sum () {
```

```
    int n=0;
```

```
    int sum=0;
```

```
    while (n >= 0) {
```

```
        sum += n;
```

```
        cin >> n;
```

}

```
    return sum;
```

}

Ques Print odd till n

```
#include <iostream>
using namespace std;
void odd ( int n ) {
    for ( int i = 1; i <= n; i++ ) {
        if ( i % 2 == 0 ) {
            continue;
        }
    }
}
```

Ques Check prime number

```
bool prime_check ( int n ) {
    for ( int i = 2; i <= sqrt ( n ); i++ ) {
        if ( n % i == 0 ) {
            return false;
        }
    }
    return true;
}
```

math.h needs to be included
for the function to work

16.0 Switch - Case

↳ Substitute for long if that compare a variable to multiple values.

↳ Switch is a multiway branch statements which provides an easy way to dispatch execution to different part of code, based on value.

→ No duplicate case allowed.

→ Optional break & default statement.

→ Two cases can't be same.

→ Constant expression must be used instead of variables in case.

Some interesting programs about switch-case,

16.1) Using var as case,

```
const int i = 0;
switch (in)
{
```

case i:] → Here case will

//STATEMENT work as it is

break; provided constant
value.

...

}

16.2) Find largest b/w two no using switch case

```
switch (n1 < n2)
{
```

case 0:] → We use case 0 to

//STATEMENT denote false case

break;

~~default:~~

] → We use default because
//STATEMENT everything except 0

break; is true.

}

Ques

Simple Calculator

```
int calc (int n1, int n2, char op){
```

```
switch (op) {
```

case '+':

return n1 + n2;

case '-':

return n1 - n2;

case '*':

return n1 * n2;

case '/':

return n1 / n2;