Lecture - 3

9.00

Flowchart / Pseudo Code → C++ Code → Compiler → Machine Language Code

10.00

11.00

Compiler - It is a piece of software that converts high level language to machine language code so that computer can actually execute it. Also it do error checking of high level code and perform optimization.

12.00

1.00

Memory chart

2.00

all
These are in power of 2 because a transistor can store only two values 1 or 0

1 bit = Binary
8 bit = 1 byte

3.00

(2^{10}) 1024 bytes = 1 KB
(2^{10}) 1024 KB = 1 MB

4.00

1024 MB = 1 GB
2^{10} GB = 1 TB

5.00

2^{10} TB = 1 PB

6.00

→ How characters are represented
→ For this we have ASCII (American Standard Code for Information Interchange), which have assigned characters, some numeric values so, ASCII no of a particular character converts to binary.

7.00

Notes This ASCII standard is globally accepted so that communication worldwide can happen without any conflict.

11

Tuesday

February 2020

Week 07

Day 042 - 324

Date 11 - 02 - 2020

FEB 20

M	T	W	T	F	S	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	

* BASIC STRUCTURE OF C++ CODE

9.00

```
int main ()
{
    CODE;
}
```

10.00

11.00

std:: cout <<" Hello krishna"; → code to print something on screen, Here "Hello krishna", but here we need to first add header file #include <iostream>

12.00

Here we can shorten code.

1.00

```
#include<iostream>
using namespace std;
int main()
{
  cout << Hello World;
}
```

2.00

3.00

→ Here we didn't wrote syntax std:: cout<< ; because ininitially we declare namespace std so whenever it will see cout, it will check cout code in std namespace auto matically.

4.00

5.00

6.00

7.00

* Variables

→ Variable are containers for storing data values.

* Data Types

Notes

Data types tells us which type of data it is either it is number or it is text or any other type. All variable uses data type during declaration to restrict the type of data to be store.

C++ Supports following data types

1.) Primary or Built-in or Fundamental Data type
2.) Derived data types.
3.) User defined data types.

Datatypes in C++

Primary	Derived	User Defined
- Integer	- Function	- class
- character	- Array	- Structure
- Boolean	- Pointer	- Union
- Floating point	- Refference	- Enum
- Double Floating point		- Typedef
- Void		
- Wide Character		

* Declaring variable

data type variable_name;

eg → int a;

Now what will happen since integer takes 4 bytes of memory, then block of 4bytes in memory is reserved & its mapping will be done with 'a'. By default garbage value will be stored in this block. Data is stored in binary format

→ int a = 10;

in memory → 0 0 0 0 0 0 0 0 0 ... 0 1 0 1 0

as this block is of 32 bits

Notes

13
Thursday

February 2020
Week 07
Day 044 - 322
Date 13 - 02 - 2020

FEB 20

M	T	W	T	F	S	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	

B ← A

(0 0 0 0 .... 1 0 1) block of memory
↳ 5 in integer

here we follow a standard to store values in binary from A → B

* Rules to declare variable names

1.) Names can contain letters, digits & underscores.
2.) names must begin with a letter or an underscore(_)
3.) Names are case sensitive.
4.) Names cannot contain whitespaces or special characters like !, #, %. etc.
5.) Reserved words (like c++ keywords, such as int), cannot be used as names.

How much memory data types store.

char - 1byte	short - 2bytes
int - 4byte	long - 8 bytes
float - 4 bytes	long long - 8bytes
double - 8 bytes	bool - 1byte

* Storing -ve numbers in memory

→ Now since a particular numeric data type will be allocated fixed no of memory for eg → lets take case of integer - 32

so here integer can store 32 bits so maximum possible numbers of different values are 2^{32}. The scientists decides to distribute this value half to negative

Notes

in 32 bit: 2^{31} | 2^{31}

$-1, -2 \cdots -2^{31}$ | $0, 1, \cdots 2^{31}-1$

numbers & half to positive numbers. Then they come out with an idea that MSB bit (most significant bit) (leftmost bit) → let it represent sign of number

0 → +ve
1 → -ve

so to represent 2 in binary it was like 0 1 0 (sign ↵, → number)

to represent -2 in binary 1 1 0 (negative ↵, → number)

but then came a problem to represent 0
as 0 0 0 was equal to 1 0 0
which was not necessary.

Then better solution came → to store -ve no
-2 → first simply convert into binary 2 - 0 1 0
and then take 2's complement & then store it.

1's complement 1 0 1
2's + 1
1 1 0 → this is how no will be stored

→ to read -ve no again do 2's complement

→ 1 1 0 → 1's complement 0 0 1
+ 1
0 1 0

now since sign before conversion was -ve as MSB was 1 then answer will become [-2]

Notes