How To Master coding

DAY 11
21/07/2025

important things to Master coding

1. Learn Basics
2. Practice EveryDay, incremently work on harder problems
3. Debug
4. Run, make planned mistakes, compile & Debug
5. Do mini fun projects
6. Learn from others / Git HUB
7. Logic Building

→ understanding about computer fundamentals, computer compiler, software coding Role.

→ gradually increase the difficulty level of problem you are choosing to solve. like Very Basic to Advanced let say addition of 2 Numbers to Graph

Incrementally work on Harder problems

$$\begin{tikzcd} \text{Basic} \arrow[r, dash] & \text{Easy} \arrow[r, dash] & \text{Medium} \arrow[r, dash] & \text{Hard} \arrow[r] & \text{Harder} \end{tikzcd}$$

Practice problem solving Everyday in computer

3 Debugging

The Person who masters DeBugging He will be a Good Engineer.
can solve Hard problems Easily using DeBugging.

Debugging is an Important skill of computer science & coding.
used to fix Bugs

4. Run your code and make some planned mistakes on your code Removing any "" or ; or commenting any line or Renaming any Variable name ent.

   make planned mistakes and let see what Happens to the code. Debug and fix it

5. Do some mini projects on learned Topics

6. Learn from others / Git

   as an Engineer have to Be ready to learn life long in the coding Journey

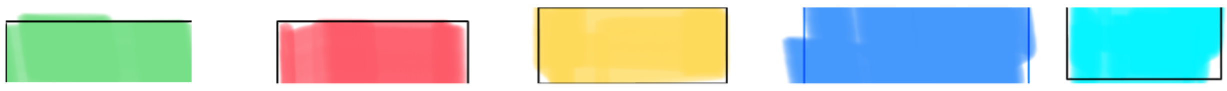
   Have to see others code, like How other's Built amazing software with the code. learn from their code

7. Logic Building

   To Improve Logic Building In coding we Must Have the ability to write Algorithms in form of code & Application of the Algorithms.

   Divide and Conquerer method's

lets understand with a simple Example



Search for color Blue

step By step process to Search for color Blue

1. Traverse one item at a time
2. Compare the color == Blue

if True return Found → True

③ else go to Search on next element

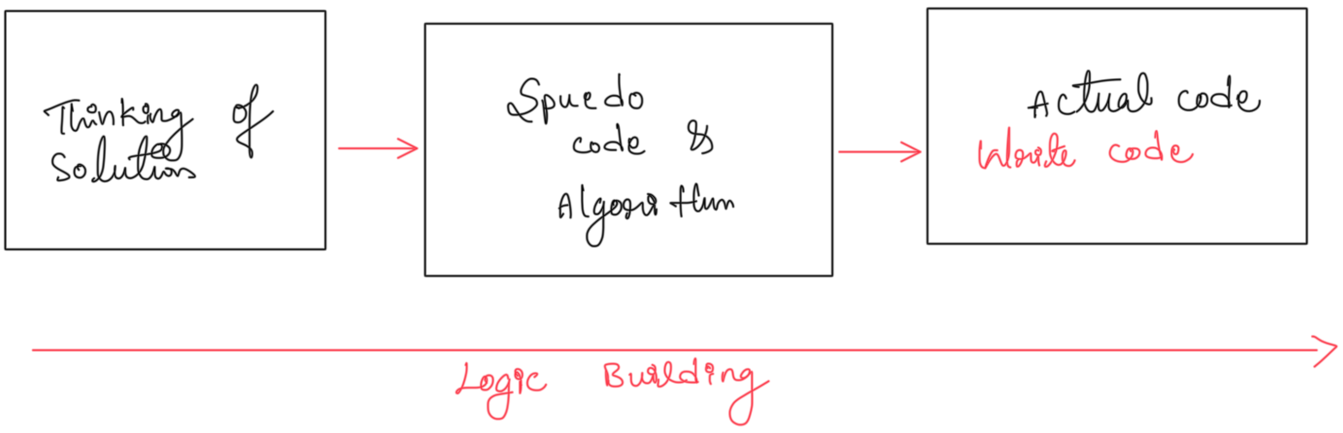
④ Repeat till traversal complete

Spuedo code

```
① Iterate over color list / Array        → Looping
② Check if item is green
        → use of "if statement"
③ Otherwise continue to next element
```

→ Algorithm series of steps to solve the problem

Translate this psuedo code to Actual code By seeing the Algorithm steps and understanding Requirements.



finding Highest lowest element, comparision's
Practice Searching, sorting, Insertion, deletion Algorithms in the Begining.

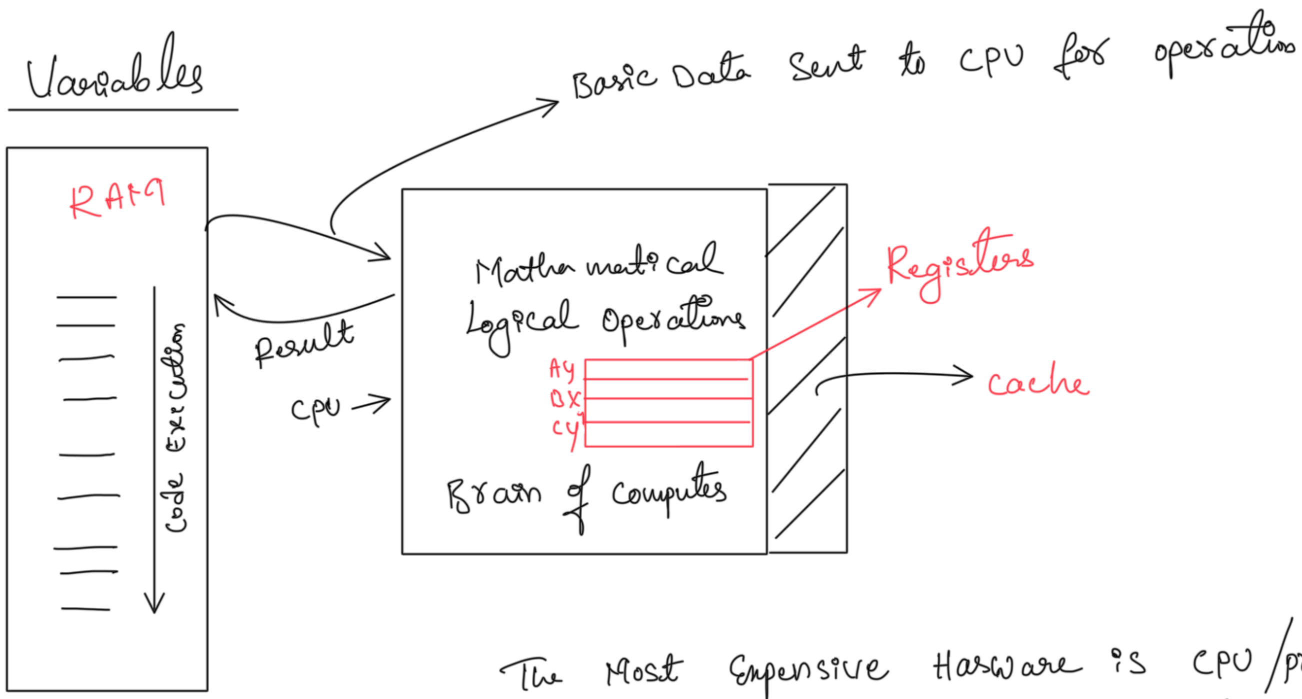
Ability of Remembering Process & Technique
Learn about others Data structures

Variables & Data Types

one of the foundational lesson "Variables & Data types"

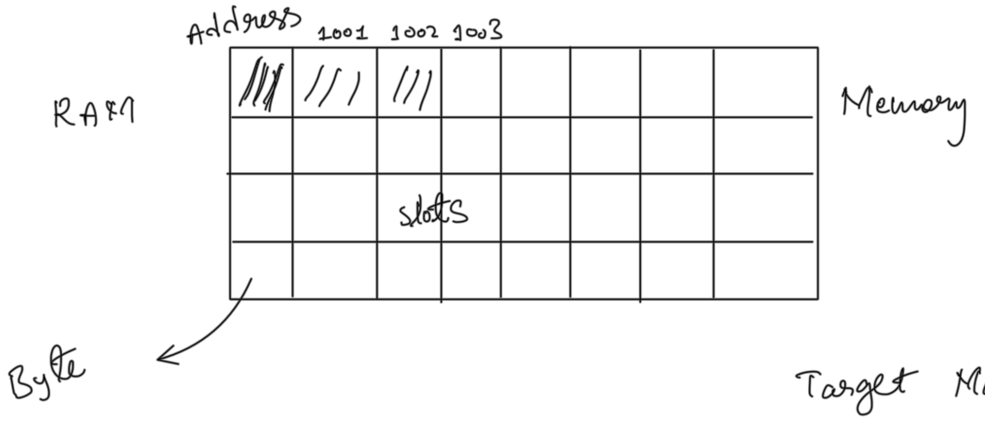
Why? Need of Variables & Data types

Variables



The Most Expensive Hardware is CPU/processor
So We Needed RAM for cost cutting.

We Load software to the RAM
We Access Software from the RAM through processor.



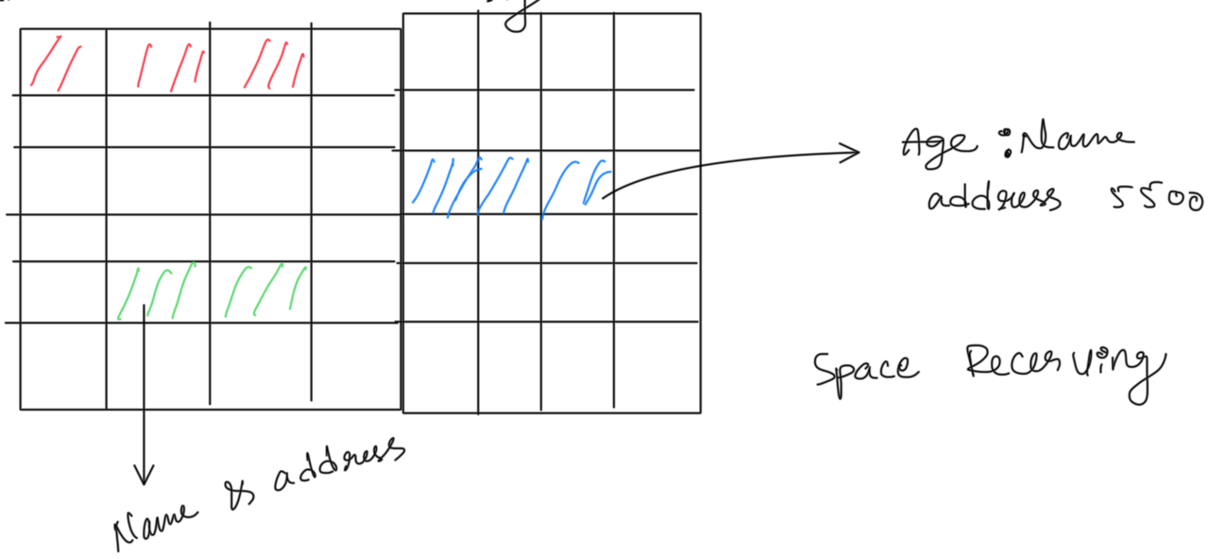
Target Machine RAM capacity is unknown, so the Variables Came into picture to develop software without thinking about Target Machine

Operating System Allocate Memory Based on the software

Requirement.   RAM Capacity

programmers dont need to remember the address where the data is stored in the Memory, He can Just remember the Name given to a address.

Yes Variable is Nothing But a Name Given to address in the Memory



Why Variables Exists ?

Coding would Have More Difficult if Variables have not used, BCZ programmes should have to remember the memory address where Data is stored But the Variable do that Job for the programmes which makes Data storing, memory allocation Retrieving of the Data Easier.

① simplicity $\longrightarrow$ Makes code More Readable

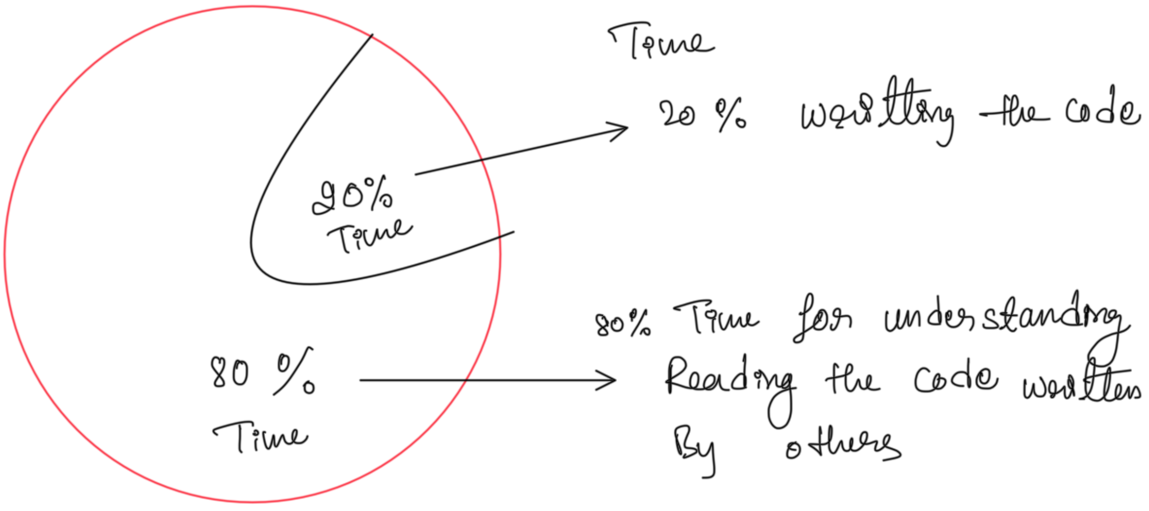
③ Flexibility of allocation Dynamic Memory Allocation

Rules to declare Variable Names

① We cannot use keywords
Example (int for while class)

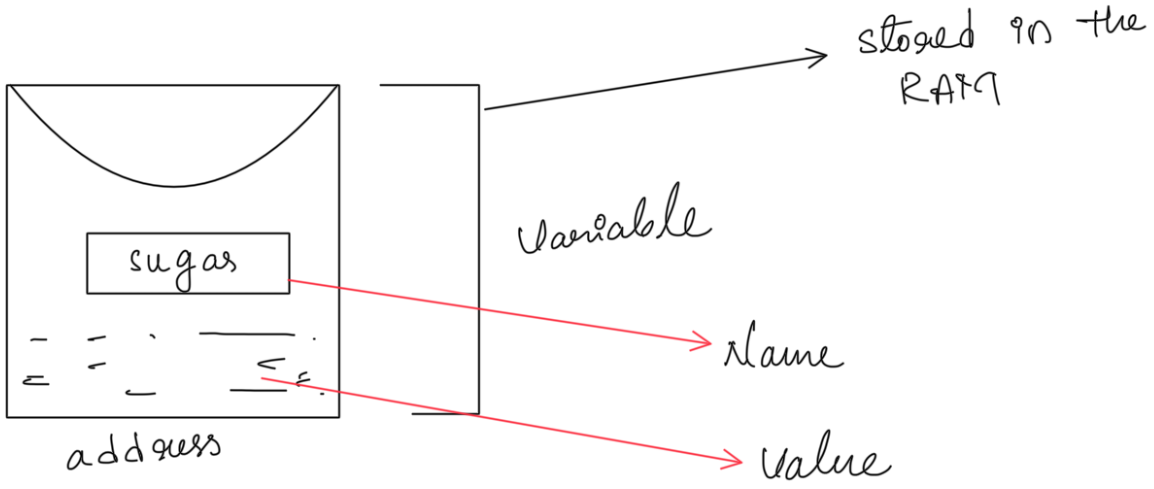
② [illegible]

(2) hive appropriate Names to Variables based on the use case and the Data it is storing.



So keep the Variable names Simple Readable

It is for the Humans We Have to make the code more Readable understandable with the help of Variable



DATA Types

Why need of DATA Types ?

Age → 0 to 100

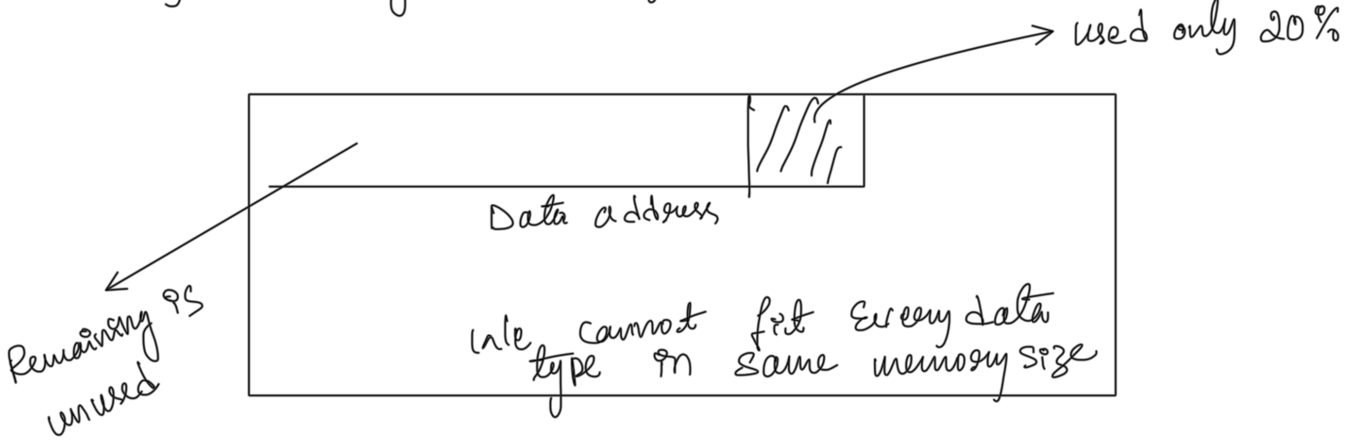
standard → 1 to 12

Name → A to Z a to z characters

Height → decimal value.

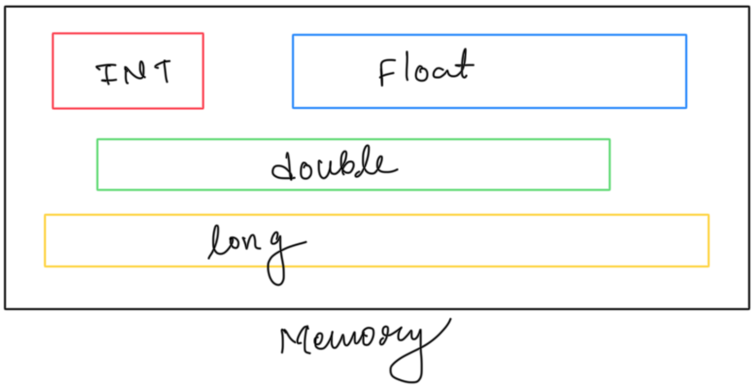
Data types have fixed size.

Why Data types have fixed size.



Data overflow or Data loss (compression)

To avoid this
So Each data type have some particular size



Based on the Data there are different Datatypes

Types of Data types

Data + Type

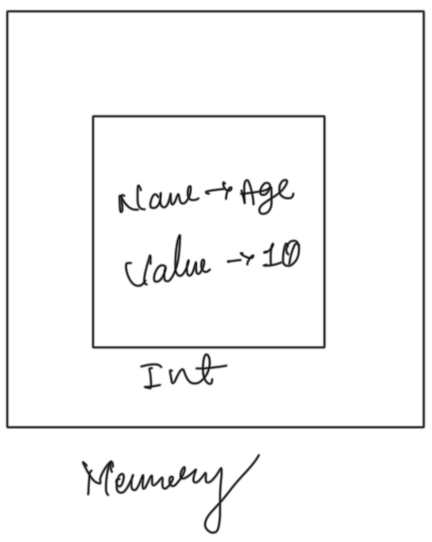
↓ ↓

to kind of Informatio

Information [illegible]

int age = 10 ;

Data type — Name of Variable — Value.



① Data types are Introduced for Efficient use of RAM space.

② Type checking (cannot store another type of Data in some other type of Data)

③ Type Casting (converting Data from one form to another)

$$\underset{\text{String}}{\text{"25"}} \longrightarrow \text{Int}$$

④ Data fetching Become Easier

⑤ Reservation of Memory to Execute program

Data type
- bool
- char
- unsigned char
- int8_t
- uint8_t
- int16_t
- uint16_t
- short

unsigned short int
unsigned int
int32_t
uint32_t
float
long
unsigned long
int64_t
uint64_t
double
long double

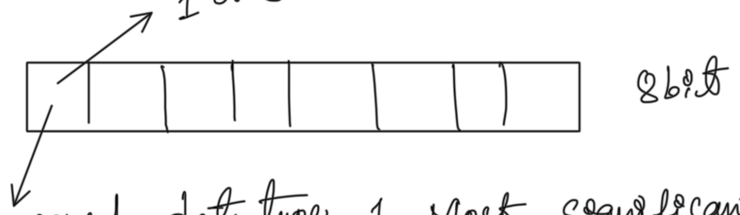
Different types of Data

Example

8bit → 1Byte

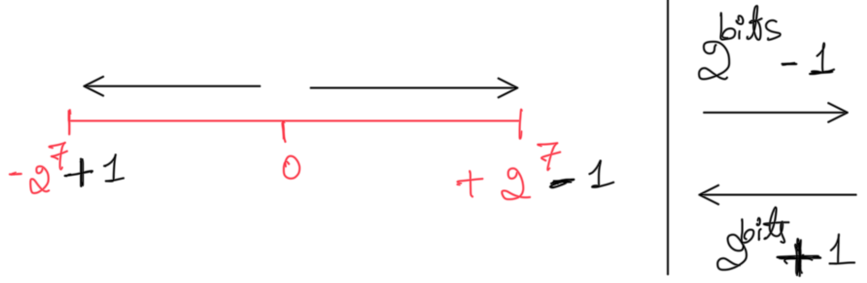
1 or 0 -ve or +ve

signed

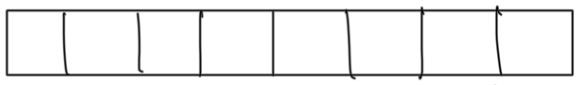


in signed data type 1 Most significant Bit is used for storing the sign

Numbers scale



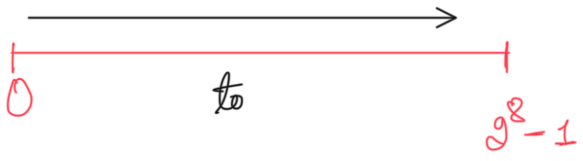
unsigned char



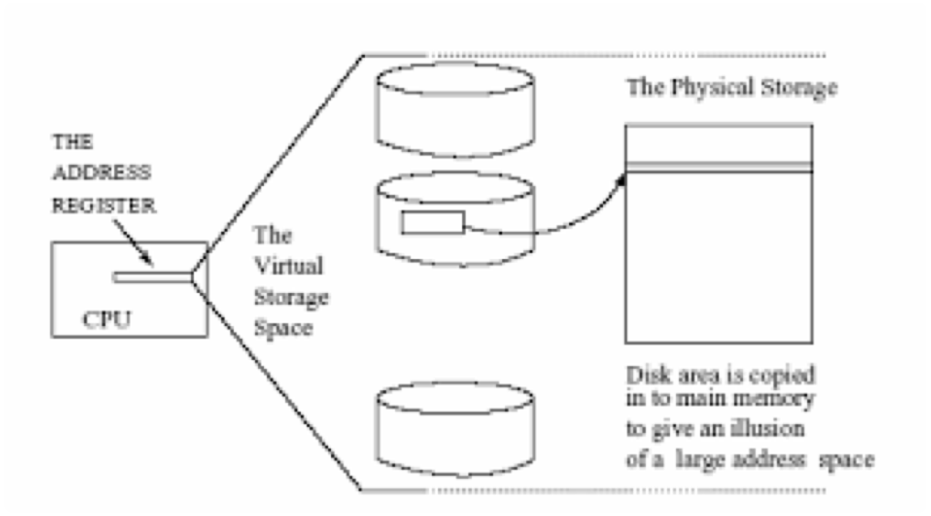
NO MSB reservation

8bits are used to store the value

Numbers scale



using sufficient data storage space for cost cutting in the Application Execution and Save memory.


THE ADDRESS REGISTER
CPU
The Virtual Storage Space
The Physical Storage
Disk area is copied in to main memory to give an illusion of a large address space

Data Type	Min Value	Max Value	Number of Bits
bool	0 (false)	1 (true)	1
char	-128	127	8
unsigned char	0	255	8
int8_t	-128	127	8
uint8_t	0	255	8
int16_t	-32,768	32,767	16
uint16_t	0	65,535	16
short	-32,768	32,767	16
unsigned short	0	65,535	16
int	-2,147,483,648	2,147,483,647	32
unsigned int	0	4,294,967,295	32
int32_t	-2,147,483,648	2,147,483,647	32
uint32_t	0	4,294,967,295	32
float	~1.4E-45 (smallest positive value)	~3.4E+38 (largest positive value)	32
long	-9,223,372,036,854,775,808	9,223,372,036,854,775,807	64
unsigned long	0	18,446,744,073,709,551,615	64
int64_t	-9,223,372,036,854,775,808	9,223,372,036,854,775,807	64
uint64_t	0	18,446,744,073,709,551,615	64
double	~4.9E-324 (smallest positive value)	~1.8E+308 (largest positive value)	64
long double	~3.4E-4932 (smallest positive value)	~1.1E+4932 (largest positive value)	80, 96, or 128

Programming Languages Supported
C, C++, C#, Java, Python
C, C++, Java
C, C++
C, C++
C, C++
C, C++
C, C++
C, C++, Java
C, C++
C, C++, Java, Python
C, C++
C, C++
C, C++
C, C++, Java, Python
C, C++, Java
C, C++
C, C++
C, C++
C, C++, Java, Python
C, C++