Programming Fundamentals

→ Thought Process to solve a Problem
- understand the problem
- input values
- approach

Problem → Solution in Head → Rough Solution / Flowchart / Pseudo Code → High level language (code) → Machine Understandable language (binary 0's & 1's)

→ Algorithm

High level language (code): User friendly language (C/C++)

Machine Understandable language (binary 0's & 1's): Computer understands only this

Architect → Design
Labourers → Work

similarly

Programmers → Algorithms
Majdoor → Code

→ Flowchart

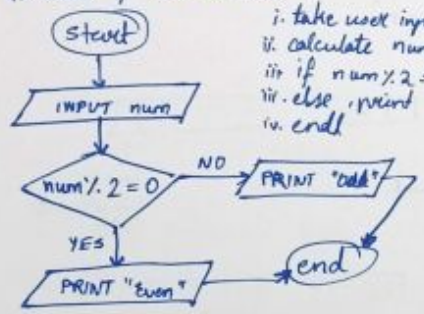
- start / end ← terminator
- read / print ← input / output
- sum = a+b ← process block
- NO / YES ← Decision-making Block
- ↓ ← flow

→ Pseudo Code (write anyhow)

Q) difference of 2 numbers
i. read a & b
ii. calculate diff = a-b
iii. print diff

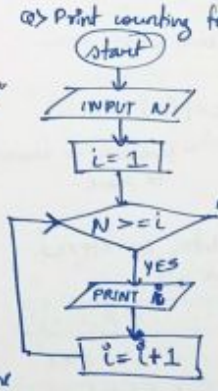
Q) avg of 2 no's
i. start
ii. cin >> a >> b;
iii. int integer value of avg variable be equal to $\frac{a+b}{2}$
iii. printf avg
iv. end

Q) check if even or odd



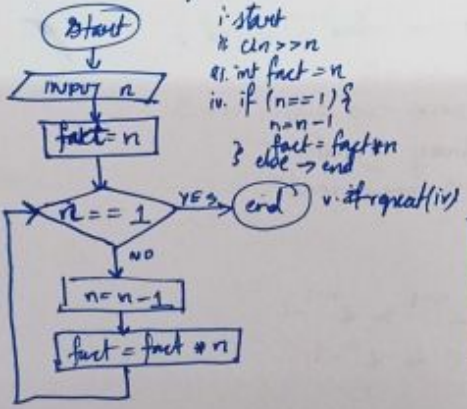
i. take user input for num
ii. calculate num % 2 -
iii. if num % 2 == 0, then print "Even"
iii. else, print "Odd"
iv. endl

Q) Print counting from N to 1



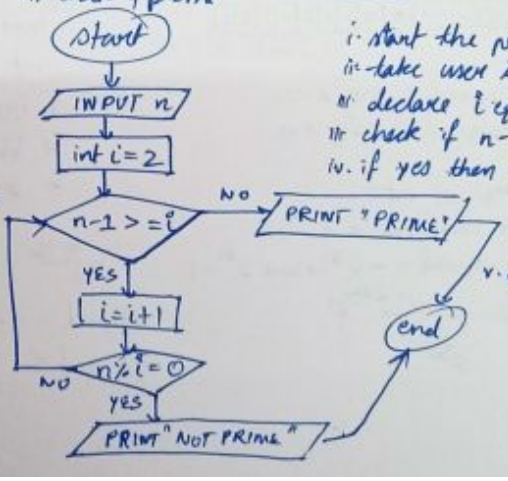
i. start
ii. take user input N
iii. define i = 1
iv. check if N >= i
v. if true then print i & increase value of i by 1, again check v. (repeat (v))
vi. else end

Q) Factorial of number



i. start
ii. cin >> n
iii. int fact = n
iv. if (n == 1) {
n = n-1
fact = fact * n
} else → end
v. if repeat (iv)

Q) check if prime

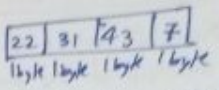


i. start the programm
ii. take user input n
iii. declare i equal to 2
iv. check if n-1 >= i
v. if yes then do i = i+1 & check n%i = 0
if this true then repeat (iii)
else print "NOT prime" & end
vi. else if (iv). print "PRIME" & end

→ Compilation Process

C++
```
#include <iostream>
using namespace std;
int main(){
    cout<<"Hello World !"<<endl;
    return 0;
}
```

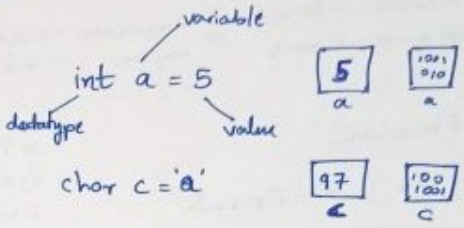
C
```
#include <stdio.h>
int main(){
    printf("Hello World !\n");
    return 0;
}
```



- compiler knows which & how much data to access with help of data type

→ IDE / Code Editor
eg. VS code, sublime text, replit

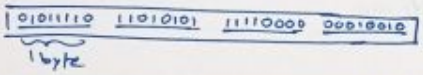
→ Datatypes (helps in → type of data stored, → necessary space)
- built-in / primitive → int, float, void, etc
- Derived → arrays, pointers, references
- User-Defined → structures, unions, classes, enumerations



char → 1 byte = 2^8-1 values
short → 2 bytes = 2^{16}-1 "
int → 4 bytes = 2^{32}-1 "
long → 8 bytes = 2^{64}-1 "
float → 4 bytes = 2^{32}-1 "
double → 8 bytes = 2^{64}-1 "

1 byte = 8 bits
bits → 0 1

int → 4 bytes = 32 bits

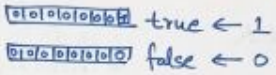


↳ $2 \times 2 \times \dots$
↳ 2^{32} combinations

1 - 1
2 - 10
3 - 11 ← 2 bit max no.
4 - 100
5 - 101
6 - 110
7 - 111 ← 3 bit max no.
8 - 1000
9 - 1001
10 - 1010

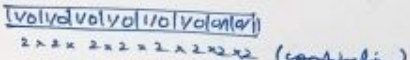
sizeof(a)
↳ function to show bytes taken to store

bool flag = true;



memory 1 byte coz can't assign smaller memory than this

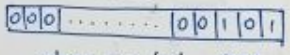
char → 1 byte = 8 bits



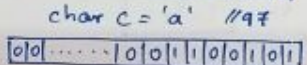
↳ 2^8 combinations

char → Ascii value
a → 97
b → 98

→ How data is stored
int a = 5



char c = 'a' //97



1's complement
- flip the rest 0's & 1's
  0 = 1
  1 = 0

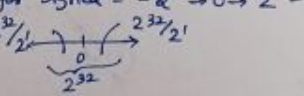
0000 0101
= 1111 1010

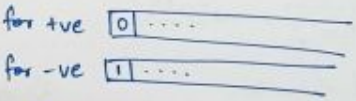
2's complement
- add 1

1111 1010
+1
= 1111 1011

Signed = +ve & -ve
unsigned = only +ve

- Range of int
  for unsigned = $2^{32}-1$
  for signed = $-2^{31} \to 0 \to 2^{31}-1$





how -ve value is stored ?
int a = -5
i. ignore -ve sign ; a = 5
ii. binary equivalent
iii. 2's complement

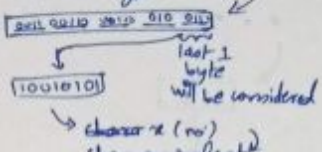
n bits
↓
signed = -2^{n-1} to $2^{n-1}-1$
unsigned = 0 to 2^n-1

→ Type Casting / Type Conversation
- conversation of one data type to another in a program either automatically by compiler (implicit type conversion) manually by programmer (explicit type casting)

implicit : char c=97;
cout<<c; // a
int c='b';
cout<<c; //98

explicit : float a=(float)2; //2.000
// variable = (type) expression;

char ch = 234342
// will not give error



↳ character (no) char equivalent

int i=1
for(; ;){
if (i<=n){
// body
i++
}
}

int / int = int

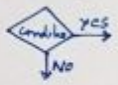
float / int = float

double / int = double

int / float = float

→ Operators
- Arithmetic → +, -, *, /, %
- Relational → >, <, >=, <=, !=, ==
- Assignment → =
- logical → &&, ||, !
- Miscellaneous → Ternary operator → condition ? x : y (if — then — else —)

→ Conditionals → loops

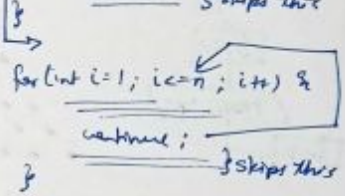


- for (int i=1; i<=n; i++){
  // body
  }
- ~~while (i<=n)~~
- i=1
  while (i<=n){
  // body
  i++;
  }
- i=1
  do{
  // body
  i++;
  }while (i<=n)

terminate the loop

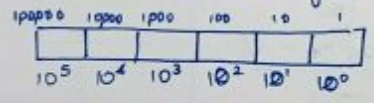
→ Break & Continue
↓
skip to the next iteration of the loop

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

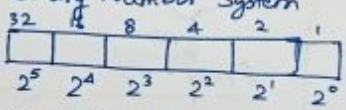


→ switch (condition){
case 'a':
cout<< "A"<<endl; break;
case 'b':
cout<< "B"<<endl; break;
case 'c':
cout<< "C"<<endl; break;
default:
cout<< "zzz"<<endl; break;
}

Decimal Number System ↔ Binary Number System



$1234 = 1\times10^3 + 2\times10^2 + 3\times10^1 + 4\times10^0$

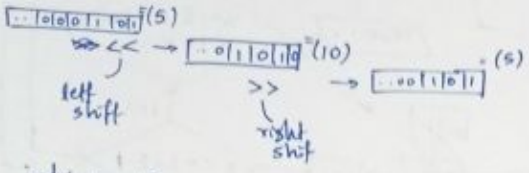


$45 = 1\times2^5 + 0\times2^4 + 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$
↳ 101101 (in binary)

→ Bitwise Operators
- AND &
- OR |
- NOT ~
- XOR ^

- left & Right Shift Operators

<< , >>




```
int a= 12;
a= a << 3;
cout << a << endl  // 96
a= a<<1 ;
```

```
int a= 12;
a = a<< 1 ;    // 12x2=24
int a= 12;
a = a<< 2;  // 12 x 2 x 2 = 48
int a= 12;
a= a<< 3;   // 12 x 2x2x2 = 96
```

↳ a = a << n // a x 2^n

↳ a = a >> n // a / 2^n , cathes → negative values
high value +ve numbers

→ Functions

return_type function_name ( ) { } function body
input parameters

→ Function Call Stack

LIFO

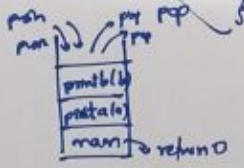
~~Pass by value - copy create value~~

```
int main (){
  int a=5;
  printa (a);
  return 0;
}
```

```
void printa(int a){
  int b=3;
  cout<< a;
  printb (b)
}
```

```
void printb(int b){
  cout<< b;
}
```

push, push, pop, pop



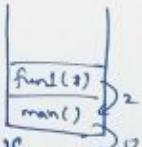
→ Pre/Post → Increment / Decrement Operator
- a++
- ++a
- A--
- --A

→ Operator Precedence (no need to learn)

→ Variable Scoping
- global variable
{
- local variable
}

```
int main (){
  return 0;
}
```

why? successfull execution of main function



Pass by Value - copy create value, not the actual value

```
main(){
  int a=5
  printNum (a);
}
```

```
printNum (int x){
  cout<< x;
}
```

```
#include <iostream>
using namespace std;
int sum (int num1, int num2);  // function declaration
int main (){
    int a,b;
    cin>>a >>b;
    int result = sum(a,b); // function calling
    return 0;
}

int sum (int num1, int num2){  // function defining
    int fvalue = num1+num2;
    return fvalue;
}
```