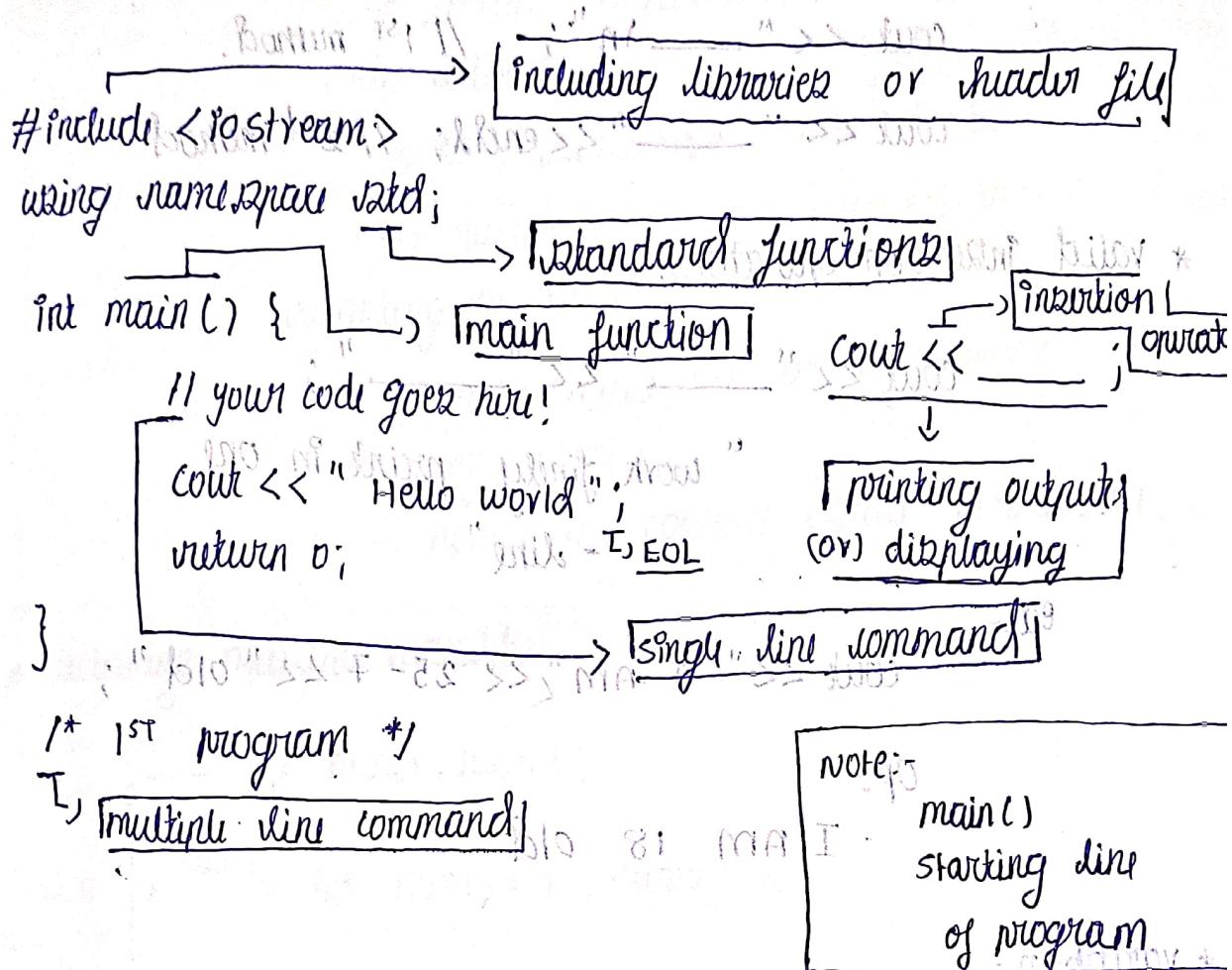


C++ programming



* Escape Sequence:-

- \n → new line
- \t → space separate or tab separate
- \\" → To print double quotes in print statement.
- \a → producing sound (carriage)

Note:-

if NOT "using namespace std;" on header file

before

so, ~~cout << "Hello world";~~ // not work raise error

std::cout << "Hello world"; // work finely

must mention if std is imported.

* moving 2nd line: ~~cout << age;~~

cout << " " << endl; // 1st method.

cout << " " << endl; // 2nd method

* valid insertion operators:

cout << " " << endl; // work finely print in one line

cout << "I AM " << 25 << " old";

option I AM 18 old

* variables:

↳ basic container to store data

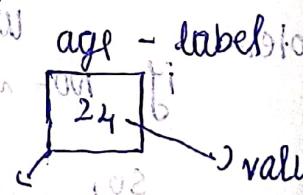
→ labeling a name to variable is known as identifier

formatting of storage unit of "

int age; // declaration

age=24; // initialisation or assign a value

data types <integer> ^{data storage place} _{value}



void main() {
 int age=25; // declaration +

printf("%d\n", age); // initialisation,

cout << age; // display if next op is 25

↳ display variable

- * rules of naming variables:
 - can contain letters, numbers, underscore
 - only allowed.
 - case sensitive $\rightarrow \text{ABC} \neq \text{abc}$
 - numbers are not allowed in the starting of the variable.
- name must not be key word (special meaning word)
- name not contain space in-between.

* declaring multiple variables:

notebook:-
int num1, num2;

also,
also, int num1 = 1, num2 = 2;

also,

$\rightarrow \text{num1} = \text{num2} = 9;$

* constant:-

ex:-

const int dob = 2000;

variable can't change [we can't change]

if try raise error as "read only variable"

* data types:- & Input:-

int age;

\downarrow cin > age;

extraction operator

* float = decimal point [5 digits precision] \rightarrow double

* double = decimal point [store digit above 5 precision]

Ex:-

float pi = 3.1415161478;

cout << pi;

OP:-

Wanted Output
3.141512

\hookrightarrow round off to 5 digit precision.

Required in ways

double pi = 3.1415161478;

cout << fixed << setprecision(10); } ->

cout << pi;

OP:-

3.1415161478, i = 1 must

To setprecision
is must to
print all digits

must give

#include <iomanip>

for w21 setprecision function.

* bool - true or false

print ex:- using cout
bool isadult = true;

cout << isadult;

OP:-

"true" \rightarrow true for true
 \rightarrow for true

printing newline



* char = storing single character data type.

	[ASCII value]
char gender = 'M';	char gender = 65;
cout << gender;	cout << gender;
OP:-	OP:-
M	A
	i d p = false
	i d p = false

* string (basic) = "store words para-sentences"

* `#include <string>` \Rightarrow header file.

string greeting = "Hi Hi Hi" (string)

cout << greeting; // endl; // \n

OP:-

Hi Hi Hi

i d p = mwe

- 90

word:-

INT - 4 bytes

float - 4 bytes

double - 8 bytes

bool - 1 bytes

char - 1 bytes

primary

data types

secondary ref

data types.

string - series of
char bytes

Note:- size of (variable - name)

OP:-

size display in bytes

ex:-

int age = 24;

float int gpa = 9.56;

Low level size of (age);

Low level size of (gpa);

subarray

i d p = subarray

DP:-

4 \rightarrow int size

4 \rightarrow float size



* conversion: - $a = 6.7$; // implicit conversion. diff

$\rightarrow \text{int } a = 6.7$; // implicit conversion.

OP:-

6

$1110 = 1110$ $\rightarrow 6.8 = 1110$

$0110 = 0110$

$0110 = 0110$

\rightarrow also;

$\text{float } b = 6.8;$

$\text{cout} \ll \text{float}(b);$ // explicit conversion.

OP:-

6

* Increment & decrement:

" $\text{int } a = 15;$ " $i(\text{op}) < \text{op}$ \rightarrow 15
" $\text{float } a = 0;$ " $i(\text{op}) > \text{op}$ \rightarrow $15 \rightarrow 16$
 $\text{a}++;$ $i(\text{op}) > \text{op}$ \rightarrow $15 \rightarrow 16$
 $\text{cout} \ll a;$ $i(\text{op}) = \text{op}$ \rightarrow $15 \rightarrow 16$

also;

[Tour, 80] OP(A) - $\text{increment/decrement}$
 $a--;$ $i(\text{op}) < \text{op}$ \rightarrow $15 \rightarrow 16$ // decremented
 $\text{cout} \ll a;$ $i(\text{op}) = \text{op}$ \rightarrow $15 \rightarrow 16$ // decremented

Type:- \rightarrow post increment \rightarrow $(\text{Data} + 0)$ \rightarrow $\text{same for decrement}$

\rightarrow post increment



$a++;$

abstraction = 100_2 , 101_2

\rightarrow $++a;$

abstraction = 100_2

* Assignment operator :- [= , += , -= , *= , /=]

int n1; // Shows to display value of n1

n1 = 25

n1 = n1 + 1

n1 += 1;

n1 = n1 - 1

n1 -= 1;

n1 = n1 * 1

n1 = n1 / 1

short form

expansion

* Comparison operators :- [== , != , <= , >= , > , <]

int n1 = 9, n2 = 8;

op:-

cout << (n1 > n2); 1 = "true"

cout << (n1 < n2); 0 = "false"

cout << (n1 == n2); 0 = 1

cout << (n1 != n2); 1

* Logical operators :- [AND, OR, NOT]

AND, OR, NOT

a + b
operators
so, operands

* Operands for logical operator is

always (0 or 1) means (True or False)

AND, OR = two operands

NOT = one operand



Diagram or truth table representation:-

AND

I ₁	I ₂	O
1	1	1
0	1	0
1	0	0
0	0	0

OR

I ₁	I ₂	O
1	1	1
0	1	1
1	0	1
0	0	0

NOT

I ₁	O
1	0

well, AND = $\&\&$, OR = $\|$, NOT = \neg

cout << (true $\&\&$ false) ;

(no - cout << (true $\|$ false) ;

cout << (!true) ;

also,

cout << (gender == 'M' $\&\&$ age ≥ 24)

age ≥ 18);

gender = 'M' .

age = 24

OP:-

Boolean

cout << (gender == 'M' $\&\&$

age ≥ 18);

More of Boolean and now how

cout << (gender == 'M' $\|$

age == 18);

((And, Or, Not) - gender is male)

More more { ((And, Or, Not) \geq two)

((Or, And, Not) \geq two)



* Variables (basic) :-

0	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	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600	601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700	701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800	801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900	901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000	1001	1002	1003	1004	1005	1006	1007	1008	1009	10010	10011	10012	10013	10014	10015	10016	10017	10018	10019	10020	10021	10022	10023	10024	10025	10026	10027	10028	10029	10030	10031	10032	10033	10034	10035	10036	10037	10038	10039	10040	10041	10042	10043	10044	10045	10046	10047	10048	10049	10050	10051	10052	10053	10054	10055	10056	10057	10058	10059	10060	10061	10062	10063	10064	10065	10066	10067	10068	10069	10070	10071	10072	10073	10074	10075	10076	10077	10078	10079	10080	10081	10082	10083	10084	10085	10086	10087	10088	10089	10090	10091	10092	10093	10094	10095	10096	10097	10098	10099	100100	100101	100102	100103	100104	100105	100106	100107	100108	100109	100110	100111	100112	100113	100114	100115	100116	100117	100118	100119	100120	100121	100122	100123	100124	100125	100126	100127	100128	100129	100130	100131	100132	100133	100134	100135	100136	100137	100138	100139	100140	100141	100142	100143	100144	100145	100146	100147	100148	100149	100150	100151	100152	100153	100154	100155	100156	100157	100158	100159	100160	100161	100162	100163	100164	100165	100166	100167	100168	100169	100170	100171	100172	100173	100174	100175	100176	100177	100178	100179	100180	100181	100182	100183	100184	100185	100186	100187	100188	100189	100190	100191	100192	100193	100194	100195	100196	100197	100198	100199	100200	100201	100202	100203	100204	100205	100206	100207	100208	100209	100210	100211	100212	100213	100214	100215	100216	100217	100218	100219	100220	100221	100222	100223	100224	100225	100226	100227	100228	100229	100230	100231	100232	100233	100234	100235	100236	100237	100238	100239	100240	100241	100242	100243	100244	100245	100246	100247	100248	100249	100250	100251	100252	100253	100254	100255	100256	100257	100258	100259	100260	100261	100262	100263	100264	100265	100266	100267	100268	100269	100270	100271	100272	100273	100274	100275	100276	100277	100278	100279	100280	100281	100282	100283	100284	100285	100286	100287	100288	100289	100290	100291	100292	100293	100294	100295	100296	100297	100298	100299	100300	100301	100302	
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--

-> accessing of strings:

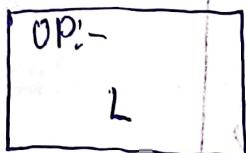
Ex CS FOUNDATION

String name = "LOKI";

[int rot, len int] : 3000

L	O	K	I
-2	-3	-2	-1

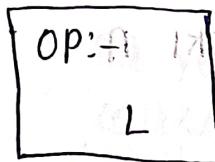
cout << name[0];



cout << name[0];

also,

cout << name.at(0);



; cout.print

"char value" >> cout

(cout.print) will be

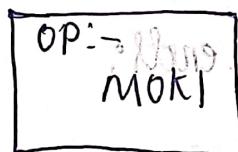
-> updating a strings:-

String name = "LOKI";

name[0] = 'M';

→ singl quoted = 'M'.

cout << name;



-> getting string from user input

String name;

cout << "enter ur name : ";

cin >> name;

cout << name;

→ This is "Lokanath" word >> cout

OP:-

Enter ur name :

LOKI

endl

{

space after
ignored



like: [applicable for int]

```
int n1, n2;
```

```
cout << "enter two num:";
```

```
cin >> n1 >> n2;
```

```
cout << n1 << endl << n2;
```

OP:- print

enter two : 23 45

23

45

-90

1

for [string]:

```
string fullname;
```

```
cout << "enter name:";
```

```
getline(cin, fullname);
```

```
cout << fullname;
```

OP:-

> (0)

enter name: LOKI A

LOKI A:90

* conditional statements :- [syntax: if (condition) { } else { }]

```
int age = 52;
```

```
if (age <= 18)
```

```
{
```

```
cout << "your child" << endl;
```

```
}
```

```
else if (age > 60)
```

```
{
```

```
cout << "your senior citizen" << endl;
```

```
}
```

```
else
```

```
{
```

```
cout << "your are an adult" << endl;
```

```
}
```



DP:-

you are an adult

40

290

100% ready mix

newspaper

posted if statements :- [year year]

example E-mail us at customers@tutor.com if you have any questions.

```
#include <iostream>
```

using namespace std;

```
int main() {
```

int year;

course "enter in year");

cin >> year;

if (year % 100 == 0) {

if (year%100 == 0) {

- [abs-fijs istanbul] \rightarrow countz "luar year";

eval {

15 = pub for
couzZ " NOT

} } (pub) 111902

else if (year%4 == 0) {

- could "map year".

9 VIP test

1609 vi

200

22

cout << "not map year"

{ "wabanbow" } >> WAB

return 0; // Standard

OP:-

enter year: 2016
leap year

OP:-

enter year: 1900
not leap year.

-190

[Every month accumulate 1 by 1]

* Ternary operators:- [if statement in one line]

(condition) ? #statement1 : #statement2;

Ex:-

int age = 14;

(age < 18) ? cout << "child" : cout << "adult";

OP:-

child

} (0 == 0010⁸ 0000) {

* switch-case [statement] [alternative of if-else] :-

example:-

int day = 2;
("Monday" 00100000)

switch (day) {

case 1:

{0 == 0010⁸ 0000} ;

case 2:

cout << "Tuesday";
break;

must give
in each
case

if break not given

in case then,

it will fall

back and execute

each case after
satisfy condition.

variables: cout << "Wednesday";
cout << "Thursday";
break;



case 4:

```
cout << "friday"; } ←  
break; { the program stops
```

case 5:

```
cout << "saturday"; } } ( ) used for  
break; { i = n step  
default position  
cout << "sunday"; } i step  
break; { ob
```

} not necessary for default,

return 0;

} { (n>i) does }

* looping:- [while, do-while, for]

→ #include <iostream> // while-loop

-using namespace std;

```
int main() {  
    int n=5; } ← initialisation  
    int i=1; } ← condition  
    while (i<=n){  
        cout << i << endl; } ← loop body  
        i++; } ← iteration - 3  
    } } ← curly braces for loop  
}; } ← curly braces for main function
```

```

→ #include <iostream> : "N, do-while
using namespace std;
int main() {
    int n=5;           initialize loop
    int i=1;           condition -1
    do {              loop body
        cout << i << endl;
        i++;           increment -2
    } while (i<=n);  condition -3
}

```

"while-loop" →* [entry control loop] →* [exit control loop]
 "do-while" →* [exit control loop]
 $i = i + 1$
 $i = i + 1$

* for-loop :-

syntax:- $i++$

$\epsilon = \text{nextFor}(\text{Initialize}; \text{condition}; \text{incrementation})$

{

// #statement to run;

}

```

#include <iostream>
using namespace std;

int main() {
    int n = 5;
    for (int i = 1; i <= n; i++) {
        cout << i << endl;
    }
    return 0;
}

```

* nested loops:-

```

int main() {
    int n;
    cout << "enter num: ";
    cin >> n;
    for (int i = 2; i < n; i++) {
        int c = 0;
        if (n % i == 0) {
            for (int j = 2; j < i; j++) {
                if (i % j == 0) {
                    c = 1;
                    break;
                }
            }
            if (c == 0) {
                cout << i << " ";
            }
        }
    }
    return 0;
}

```

* Continue & Break statements

- ↳ break statement used to Break the loop or iterations of condition.
- ↳ continue used to skip the iteration of the loop.

break:- refer nested loop program.

continue:-

```
int main()
```

```
{ int n=8;
```

```
for(int i=0; i<=8; i++)
```

```
{
```

```
    if(i%2==0)
```

```
        continue;
```

```
    else
```

```
        cout<<i<<" "
```

```
}
```

```
return 0;
```

```
}
```

```
{(0=0)}{}
```

i = 0

if(i<=8)

{(0=0)}{ }{(0=0)}{ }

{(0=0)}{}

{(0=0)}{}

{

{

{(0=0)}



* arrays :- [Elements stored in memory] = contiguous mem.

- To store similar data in one variables.

```
#include <iostream>
#include <string>
using namespace std;
```

syntax:

```
<string> variable [size] = {  
    data-type     variable  
    value, ... };  
} (colon)
```

```
int main () {
```

// same data

```
string name = "goms";
```

```
string name[5] = {"goms", "siva", "codio", "logic10",
```

```
                    "undefined"};
```

P

PF

```
string name[5] = {24, 21, 4, 1, 0};
```

```
cout << name << endl;
```

```
cout << name[2] << endl;
```

```
cout << age[2] << endl;
```

OP:-
24 -> siva

goms

0x6fffd0

0x3ffd10

0x00 -> undefined

memory starting address,

so, → accessing elements of array by indexes:

```
cout << name[0] << endl;
```

```
cout << age[0] << endl;
```

OP:-

goms - 1st element

24 - 2nd element

so,

-5 -4 -3 -2 -1 } reverse index.

goms	siva	codio	logic10	undefined
------	------	-------	---------	-----------

24	21	4	1	0
----	----	---	---	---

0 1 2 3 4

} forward index



Note:-

* size is not mandatory if declaration of initialisation takes place in same line.

* size is mandatory if declaration occur without initialisation.

→ printing of arrays :-

for (int i=0; i<5; i++)
{

 cout << names[i] << ":" << ages[i];

 cout << endl;

 cout << endl;

→ finding size of arrays:- (or) length of arrays :-

int size = sizeof(ages)/sizeof(ages[0]);

cout << size;

ex:-

$$\frac{20}{4} = 5 = \text{no. of elements}$$

Total size Size of element,



* Special for-loop for arrays:-

for (practicing) *agus*

1000 {

beautiful house could kindly

3

1. *Constitutive heterogeneity* is a feature of all living systems.

* two dimensional & multidimensional array:

```
int main() {
```

~~1D array~~
1D array
or
matrix

$\rightarrow \text{int arr2[3][3] = } \{ \{ 1, 2, 3 \},$

↓ ↓
row column
↑ ↑

$$\{ \underline{7, 8} \} \{ 9 \} \};$$

int arr[3][3][3] =

face or collection
of mut.

$\{ \quad \{ \quad \{ 1, 2, 3 \} \}, : \text{snop}$

$\{4, 5, 6\}$ \cap $\{1, 2, 3\} = \emptyset$

$\{7, 8, 9\} \cup \{7\}$

$\{ \quad \{ 10, 11, 12 \} \text{ future} \}$

-: square je integral (nicht-)ganzes $\{13, 14, 15\}$, prüfen

$\{16, 17, 18\}$

$$\text{Total profit} = \text{Total sales} - \text{Total costs}$$

* printing or display 2D array:-

bullet. no prob. follow a pattern

```
int row = sizeof(arr2) / sizeof(arr2[0]);
int col = sizeof(arr2[0]) / sizeof(arr2[0][0]);
```

for (int i=0; i<row; i++) {
 for (int j=0; j<col; j++) {
 cout << arr2[i][j] << " ";
 }
 cout << endl;
}

Output: 1 2 3
4 5 6
7 8 9

Output:

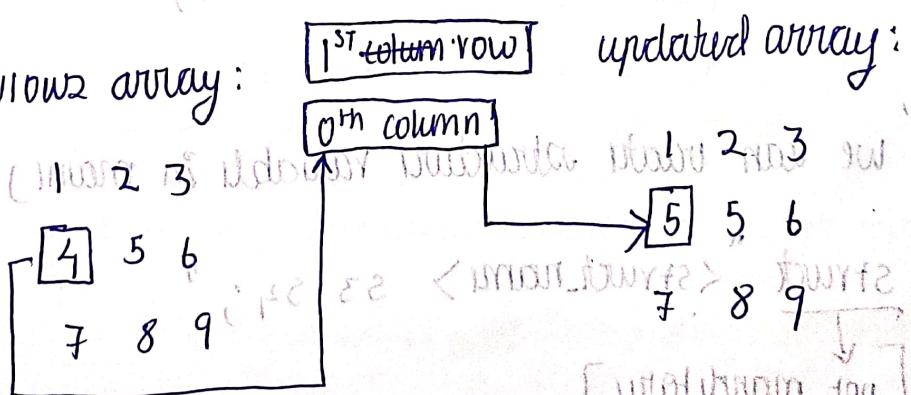
1 2 3 } matrix
4 5 6 } format display
7 8 9 }

* access & update elements in 2D arrays:-

```
arr2[1][0] = 5;
```

cout << arr2 // display array

previous array:



* **Structures**

- ↳ storing a related data is called **structures** (or **custom data type**)

```
#include <iostream>
#include <string>
using namespace std;
```

→ can specify struct name, [mandatory for different struct]

```
struct {
    string name;
    int age;
    float GPA;
}
```

} s1, s2; // structure variables.

```
int main() {
    s1.name = "Loki";
    s1.age = 18;
    s1.GPA = 8.6;
    cout << s1.name << " " << s1.age << " "
        << s1.GPA << endl;
```

X.1 we can also

also,

we can declare structure variable in main()

```
struct {
    string name;
    int age;
    float GPA;
} s3, s4;
```

[not mandatory]

* array of structures:

<struct-name> day[0][30];

* Enums:

enum Day {

 Sunday, Monday, Tuesday, Wednesday,

 Thursday,

 Friday, Saturday,

};

int main () {

 Day today = Tuesday;

 cout << today << endl;

 cout << (today + 1) << endl;

 if (today == Day::sunday)

 cout << "Holiday" << endl;

Output:

2 -> Friday

Output:

3 -> Saturday

* Pointers & References:

alias name < alias.h >

```
int main() {  
    int age = 24;  
    int aliasage = age; } // output:  
cout << aliasage << endl;  
cout << age << endl;  
cout << copyage << endl;
```

address or reference address

difference b/w copy & alias:-

```
int age = 24;  
int aliasage = age; } // output:  
int copyage = age; // output:  
age = 25;  
cout << aliasage << endl;  
cout << copyage << endl;
```

alias
reference
output: 25
alias
reference
output: 24
alias

Note:-
aliasage is not new variable
it just another name of variable age

So, variable name > address of memory location → 78H21E\$
which is address of agl

not store address of another variable, it access
address of another variable.

* Pointers:- Storing a address of another variable. Ps
called pointers.

int main() { // If you see this, it means your code failed.

- (returning) address; memory is part of backbone
int* ptr = " ", //dilating of pointer pointer
T, // (char*) can store addresses
(" : output = page, table, all func

cout << "ptr" << PTR << endl;

```
cout << "value " << * p << endl; } // output
```

* $\text{PE} \gamma = 5b$;
cout < 2agi < 2 endl;
change value by pointer.

* Dynamic memory allocation: (run time memory allocation)

```
int main() {
```

```
    int age = 24; // compile time memory allocation
```

```
    int *ptr; // dynamic memory allocation
```

```
    ptr = new int; // run time allocation
```

```
*ptr = 24;
```

```
cout << *ptr; // output
```

new - keyword

which allocates memory size of `int`:

→ creating array by dynamic memory allocation:-

```
int num;
```

```
int main() {
```

```
    int arr[num];
```

```
    cout << "enter number of students : ";
```

note:-

memory arr

stored in RAM

function

```
cin >> num;
```

```
int *arr = new int [num];
```

```
for (int i=0; i< num; i++)
```

```
{
```

```
    arr >> arr[i];
```

```
}
```

```
for (int i=0; i< num; i++)
```

```
{
```

```
    cout << arr[i] << endl;
```

```
}
```

```
return 0;
```



* functions: (code reusability)

- ↳ declaration *
- ↳ definition *
- ↳ calling *

#include <iostream>

#include <string>

using namespace std; // (nothing runs) [nothing return]

void add5(int a); // declaration

int main() {

 add5(15); // calling.

 return 0; // (nothing runs) [nothing return]

void add5(int a) {

 cout << a + 5; // definition

}

Note :-

for return any value

we must to use specific datatype

instead of void like int <function>

if return value is int



* Default arguments & call by reference

```
void add5 (int n=5) {  
    cout << n+5 << endl;  
}  
  
int main () {  
    add(15); → // O/P:- 20  
    add(); → // O/P:- 10 → default = 5  
}
```

also,

```
void add5 (string greet, int n=2) {  
    cout << greet << " " << n+5;  
}  
  
int main () {  
    add5("Hi", 6); → // O/P:- Hi 11  
    add5("Hello"); → // O/P:- Hello 7  
}
```

if :-

```
void add5(int n=2, string greet) → error because only default argument must give in last of the parameter  
// syntax error because only default argument must give in last of the parameter
```

* call by value & call by reference:-

void add5(int num)

{

 num += 5;

 cout << num << endl;

}

int main()

 int num = 6;

 function

 add5(num);

 P8

 cout << num << endl;

 return 0;

}

were, num is not changed

(Jism of

[call by reference]

if void add5(int &num)

were, num value

changed.

Then,

Output

were, if array was pass it reflect the
main() array without reference because array
was created by dynamic address. we didn't

```

void array( int arr[ ] )
{
    arr[0] = 89;
}

int main()
{
    int arr[ ] = {1, 2, 3, 4};
    array( arr );
    cout << arr[0] << endl; → {Output
    return 0; } → 89
}

```

arr[0] was changed
in main().

* Function overloading:-

```

int sum( int a, int b ) → for integer type
{
    return a+b;
}

int sum( int a, int b, int c ) → for three integers
{
    return a+b+c;
}

double sum( double a, double b ) → for floating point
{
    return a+b;
}

```

```

int main() {
    cout << sum(2, 3) << endl;
    cout << sum(2, 3, 6) << endl;
    cout << sum(3, 5, 3.5) << endl;
    return 0;
}

```

return 0;

following block is

// output:-

5

11

7.0

3 () return 3.5

* if we give another:-

```

int sum (int a, int b) {

```

return 3*a+b;

it raise error

known as

function overloading

*because, were, sum with int a & int b
are exist so, it raise error. we must give
different datatype as parameter or different
function name.*

*so we have discuss about a function
as how we avoid its reusing because we
can't do that as per
that we programmed function as follows
then we can't prove we declared new function
with same name as previous one*



* Scope of variables:-

Q1 Q2 Q3 Q4

#include <iostream>

using namespace std;

int a=5; // Global variable.

int print() {

 int a=6; // Local variable

 cout << a << endl;

}

int main() {

 print();

 cout << a << endl;

 return 0; }

Value of a is 5 at this point

so number

without return

if we comment a=6 in print()

at this point if a=6, now consider

now focus on priority

so,

trouble to remove

so, remove

#output

56

56

local variable

56

global variable

Local variable:- It's a variable which declare in

the specific function or block of code and use

only in that block code.

Global variable:- It's declared beginning of the code

which can accessible by every function, main() or entire program.



* RECURSION :- [Calling a function in the same function which work like loop concept].

example:

int fact(int n) {
 if (n == 1)
 return 1;
 else
 return n * fact(n-1);
}

flow:
 $5 \times \text{fact}(4) = 120$
 $4 \times \text{fact}(3) = 24$
 $3 \times \text{fact}(2) = 6$
 $2 \times \text{fact}(1) = 2$
 $\Rightarrow \text{return } 1$

base case - condition

function calling



Class & Objects:

OOPS = Object oriented programming

{ - Functions (methods)

{ - act on data

entity (real world objects)

main purpose - DRY [Don't Repeat yourself]

most common terms:

classes = blueprints / templates

objects = instance

class generally

contain attributes

& methods (functions)

also, access specifier.

* Declaration:

#include <string>

#include <iostream>

using namespace std;

class Student {

public: —> access specifier

int roll_no;

string name;

}; void display();

use caps letter in first
for naming class (better)

int main() {

Student s1;

s1.roll_no = 11;

s1.name = "Loki";

} s1.display();

for class
function

specify method with respect class.

```

void student :: display() {
    cout << "Roll-no: " << roll_no << "name: " <<
    name << endl;
}

```

definition of method of class in outside of the class.

* "::" scope resolution operator

* if we add another object in main program

```

student s1;
student s2;
s1.roll_no = 11;
s1.name = "LOKI";
s2.display();
s1.display();

```

>>> output:

roll_no: 11 name: LOKI
roll_no: 12 name: YUJI

* Constructors and destructors:

- special function
- called when object is created
- same name as class
- it doesn't have return type.

Three types:

- default constructor
- parameterized constructor
- copy constructor

* self - created - constructor:

```

class Car {
public:
    string brand;
    double engine;
}
Car () {
    cout << "Constructor for car" << endl;
}

```

This is self created constructor which is created by you, but if u not create constructor there will be a default constructor exist but can't.

cout << "Constructor for car" << endl;

```

int main() {
    Car swift; <<
    cout << "Car Swift, brand = " << swift.brand << endl;
    cout << "Car Swift, engine = " << swift.engine << endl;
}

```

constructor for car

- 1.2 tribute
- 1.5 i - option, 1.2
- 1.7 i - option, 1.2
- 1.0 hatchback, 1.2
- 1.0 hatchback, 1.2

* self - created - parametrised - constructor:

```

class Car {
public:
    string brand;
    double engine;
}
Car (string br, double e) {
    brand = br;
    engine = e;
}

```

cout << "Constructor for car" << endl;

brand = b;
engine = e

cout << "Constructor for car" << endl;

```

int main() {
    Car Swift("Swift", 1.2);
    cout << Swift.brand << Swift.engine;
}

```

* Copy Constructor

```

int main() {
    Car Swift("Swift", 1.2);
    Car copy = Swift;           —> Copy The Object Swift
    cout << copy.brand << endl;
}

```

so, when copy created
The copy constructor will
execute.

* Destructor

it doesn't survive or not need the argument like

```

~Car() {
    cout << "Destructor" << endl;
}

```

when we discard or when the object became
out of scope or use then, it call destructor.

class Car {
public:
 string brand;
 float engine;
 Car(string b, float e) {
 brand = b;
 engine = e;
 }
}

* This keyword:

```
class Car {
```

 Public:

```
        String brand;
```

```
        double engine;
```

```
    Car(String brand, double engine) {
```

This → brand = brand;

This → engine = engine;

```
int main() {
```

```
    Car Swift("Swift", 1.2);
```

```
class Base {
```

 Public:

```
        Base(int a) {
```

```
            cout << "Base value : " << a << endl;
```

```
};
```

```
class Child : public Base {
```

 Public:

```
        Child(int a, int b) : Base(a) {
```

```
            cout << "child value : " << b << endl;
```

```
};
```

```
int main() {
```

```
    Child obj(10, 20);
```

```
};
```

"This → brand" is
more like "object.brand"

both constructor were
called when the child
object was created.
This is the way to
name the parameters.



Pillars of OOPS:

- Encapsulation;
- Inheritance;
- Polymorphism;

Terms :- Abstraction;

* Encapsulation :- (data/attributes & function) method -

single entity - class
> (restrict access of the object)

→ public, protected, private
→ default

example:-

class Student {

 Public:

 { declared variables
 & methods are used by main()

};

 which var is public

class Student {

 { declared variables

 private: name {
 a methods without

 access specifier were default was

};

 "private"

class student {

protected:

} Inside variable
methods are protected
They can access by inherited
class not even its own object

- Bottom nature is self-inheritance - ; following
 - (class - public class)
 - (two's will be same behavior)

* Inheritance (syntax): } inherit or access a variable
and method of one class

example:

class Parent {

Public:

int age = 45;

String name = "Loki";

void display() {

cout << name << age << endl;

}

class student { } student : public parent

Public:

int age = 50;

String name = "Loki-child";

```

void display() {
    cout << "this is child" << endl;
}

};


```

```

int main() {
    parent P;
    P.display(); // prints "this is parent"
    Student S;
    S.display(); // prints "this is student"
}

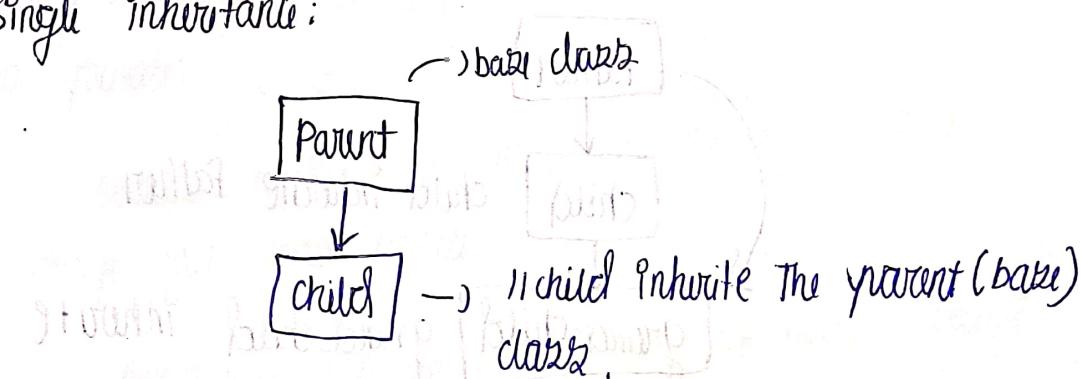

```

parent P; P.display() : prints "this is parent"
 S.display() : prints "this is student"
 Because S inherits from the base class parent

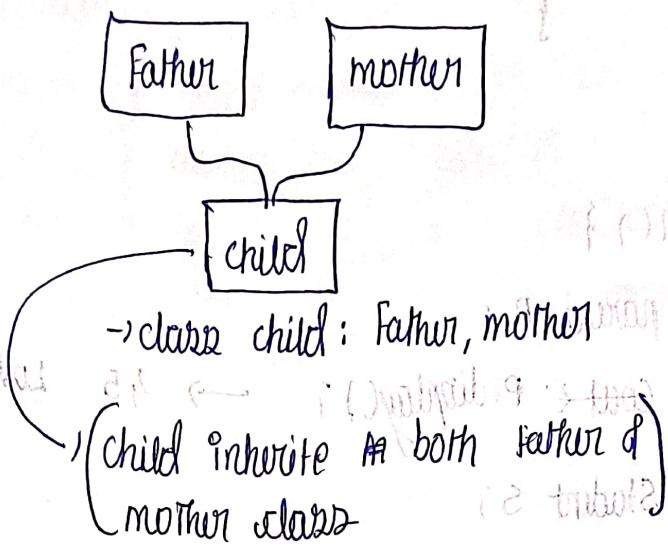
-> Types:

- single inheritance
- multiple inheritance
- multi-level inheritance
- hierarchical inheritance
- hybrid inheritance

* Single inheritance:



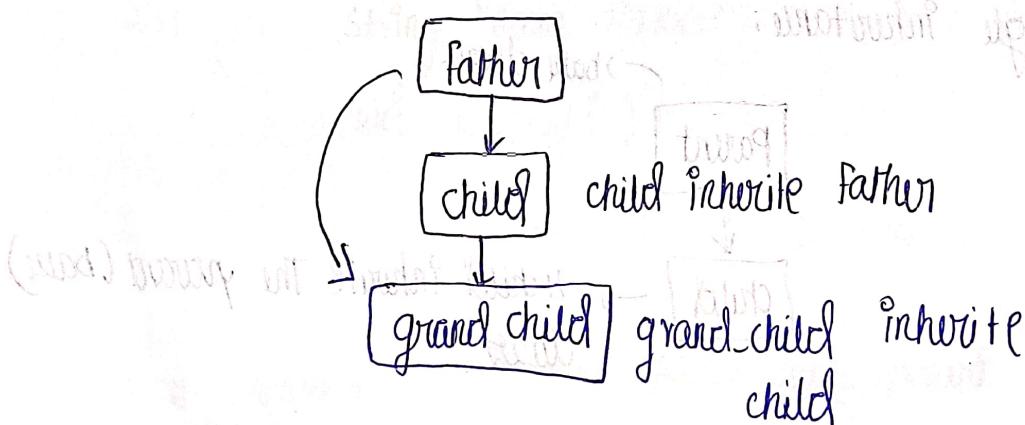
* multiple inheritance:



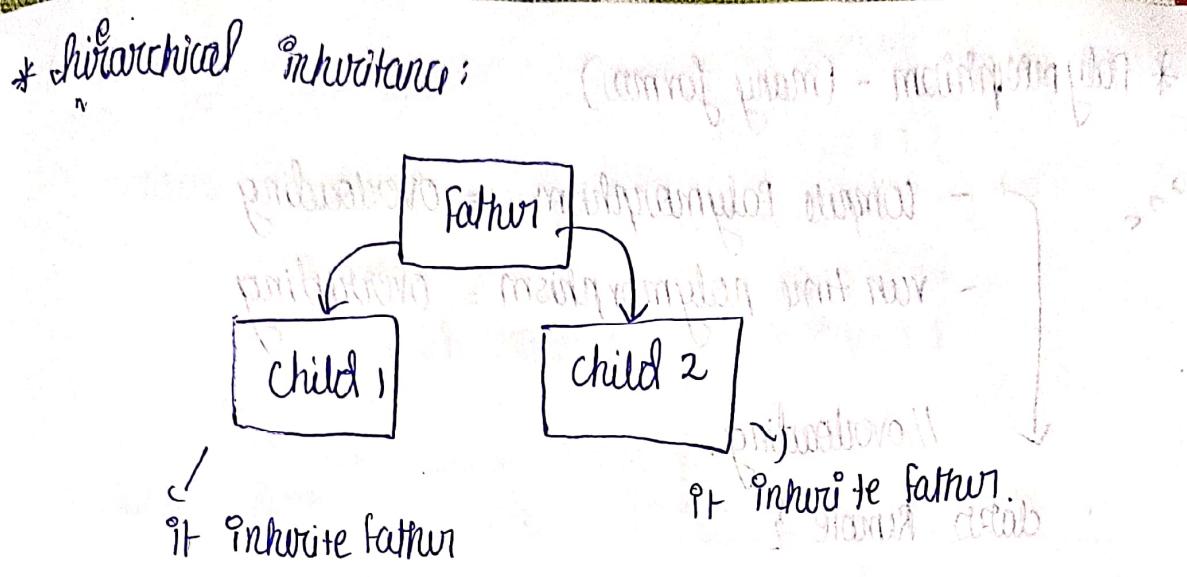
if "name" variable both present in both father & mother class if u try to access a name variable in child class. Then it raise "error ambiguous variable" so, u must specify which class variable.

ex: "Father::name"

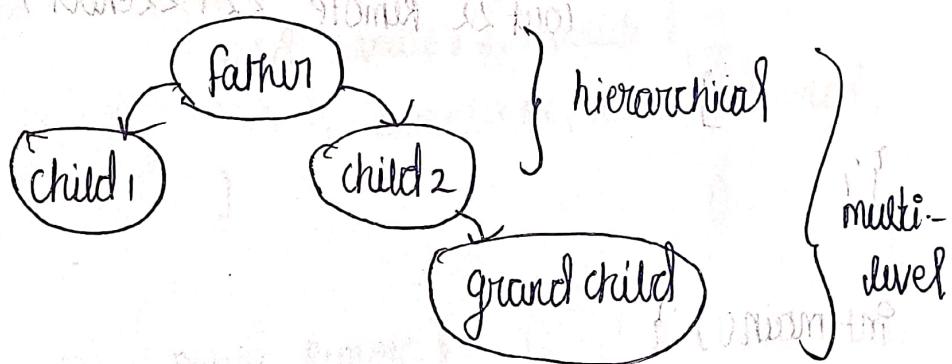
* multilevel inheritance:



so, it can also access The father



* hybrid inheritance: (can't print - handwriting)



X why base class constructor call even child inherited

as private:
because, construction is about memory + object
setup, not access control

before C++ can build child part, it must build
the base part.

* Polymorphism - (many forms)

- Compile Polymorphism = Overloading
- run time polymorphism = overriding

//overloading

class Remote {

Public

void show () {

cout << "Remote " << endl;

void show (int i) {

cout << "Remote " << i << endl;

int main () {

Remote r;

r.show(); → Remote

r.show(5); → Remote 5

// overriding.

```

class Remote {
public:
    void show();
};

class Remote {
    void show();
};

class AC : public Remote {
public:
    void show() override {
        cout << "AC Remote" << endl;
    }
};

class TV : public Remote {
public:
    void show() override {
        cout << "TV Remote" << endl;
    }
};

int main() {
    Remote *r = new AC();
    r->show();
    r = new TV();
    r->show();
    delete r;
}

```

add destructor:

```

~AC() {
    // ...
}

~TV() {
    // ...
}

```

Abstraction:-

class Shape { // abstract class

Public:

{ () virtual void area() = 0; };

}

class Rectangle : public Shape {

Public:

int width; int length;

int length;

Rectangle (int width, int length) {

This->width = width;

This->length = length;

}

void area() override {

cout << "Rectangle area : " << length * width << endl;

}

class Circle : public Shape {

Public:

int radius;

Circle (int radius) {

This->radius = radius;

}

void area() override {

cout << "Circle area : " << 3.14 *

radius * radius << endl;

}

};

```
int main() {
```

```
    Rectangle r(15, 20);
```

```
    Circle c(10);
```

```
    Shape *s1;
```

```
    s1 = &r;
```

```
    { (s1->area());
```

```
    s1 = &c;
```

```
    s1->area();
```

```
    return 0;
```

```
}
```

out:

Rectangle area: 300

Circle area: 314

X abstract class must have one : pure virtual function

input: cout

{ (Board, buns) } cout

cout << buns << endl

>> buns << " : Board " << endl

<< buns << " : buns " << endl

>> buns << endl

input: cout

Problems:

```
10 class Vehicle {  
    public:  
        string brand;  
        int speed;  
        Vehicle(int speed, string brand) {  
            this->speed = speed;  
            this->brand = brand;  
        }  
};  
  
class Car : public Vehicle {  
    public:  
        string fuel_type;  
        Car(int speed, string brand,  
             string fuel_type) :  
            Vehicle(speed, brand) {}  
        void display() {  
            cout << "Brand : " << brand <<  
                "\n Speed : " << speed <<  
                "\n fuel Type : " <<  
                fuel_type << endl;  
        }  
};
```

```
int main() {  
    cout << "TASK 1 : \n" << endl;  
    Car C1(80, "swift", "petrol");  
    C1.display();  
}  
return 0;
```

Output:

Branch: swift

Speed: 80

Fuel Type: petrol

2°

```
class Animal {
```

public:

```
void sound() {  
    cout << "Animal makes sound!" <<  
    endl;
```

```
}
```

```
};
```

```
class Dog : public Animal {
```

public:

```
void sound() {  
    cout << "Dog barks!" << endl;
```

```
}
```

```
};
```

```
class Cat : public Animal {
```

public:

```
void sound() {  
    cout << "Cat meows!" << endl;
```

```
}
```

```
};
```



Scanned with OKEN Scanner

```

int main() {
    Dog D1;
    Cat C1;
    D1.sound();
    C1.sound();
    return 0;
}

```

Output:

Dog barks!
Cat meow!

30

```

class Employee {
protected:
    string name;
    int salary;
public:
    Employee(string name, int salary) {
        This->name = name;
        This->salary = salary;
    }
    virtual void calculateBonus() = 0;
}

```

};

```

class manager : public Employee {
public:
    Manager(string name, int salary) :
        cout << "manager object" << endl;
}

```

}

```

void calculateBonus() {
    cout << name << " Bonus : " << endl;
}

class Developer : public Employee {
public:
    Developer(string name, int salary) : employee(name, salary) {
        cout << "developer objects" << endl;
    }

    void calculateBonus() {
        cout << name << " Bonus : " << salary * 0.1
        << endl;
    }
};

int main() {
    vector<Employee*> employees;
    employees.push_back(new Manager("LOKI", 500000));
    employees.push_back(new Developer("RAM", 700000));
    for (Employee* e : employees) {
        e->calculateBonus();
    }
    return 0;
}

```

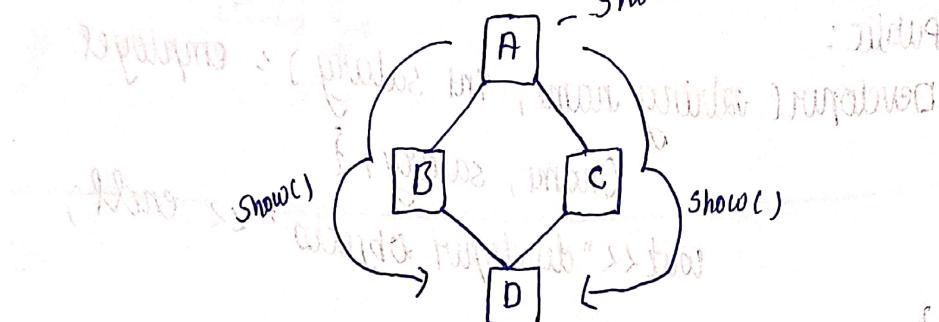
Output
Manager object
Developer object
LOKI Bonus : 50000
RAM Bonus : 70000

* Diamond problem:

consider A, B, C, D are classes

→ * B & C inherit the class A

→ * D inherits both B & C.



if The A class contain a function show()

and other B & C doesn't have function

show(). Then,

if object of D try to use show()

Then, it gets confused because, it

can only know there is two show()

one from B & other C. it raised

error or "ambiguous function show()"

This is called Diamond problem.

So, To solve this we will use virtual

while inherit A in both B & C

example:

```
#include <iostream>
#include <iomanip>
```

using namespace std;

```

class A {
public:
void show() {
    cout << "show function" << endl;
}
}

class B: virtual public A {
public:
    // Inherit A as
    // virtual class
};

class C: virtual public A {
public:
    // Inherit C as
    // virtual class
};

class D: public B, public C {
public:
    // Bi = Bio-Echt
};

int main() {
D obj;
obj.show();
return 0;
}
    }, output:
        show function
    } Acid class

```

show function

* ~~TV~~ is a remote \rightarrow inheritance relationship

Book has Author \rightarrow we can relate without inheritance

Example:

```
#include <iostream>
using namespace std;

class Author {
protected:
    int id;
    string name;
public:
    void displayAuthor() {
        cout << "Author's name: " << name;
    }
    Author(int id, string name) {
        this->id = id;
        this->name = name;
    }
};

class Book {
public:
    int book_id;
    string book_name;
    Author * author; // one class object inside in other class
};
```

```

void displayBook() {
    cout << "Book's Name: " << book_name << endl;
    author-> displayAuthor();
}

Book( int book_id, string book_name, Author *author ) {
    This->book_id = book_id;
    This->book_name = book_name;
    This->author = author;
}

};

int main() {
    Author a1(1, "JK Rowling");
    Book b1(1, "Philosopher stone", &a1);
    Book b2(2, "Chamber of secret", &a2);
    cout << b1.displayBook();
}

```

Output:

```

Book's name: Philosopher stone
Author's name: JK Rowling

```

* exception or Error handling

- > try
- > Throw
- > catch

Basic syntax:

```

try { variable = risky_code; }
      // risky code
}
catch ( type variable ) {
    // handle error
}

```

example 1:

```

#include <iostream>
using namespace std;
int main() {
    int a=10, b=0;
    try {
        if (b == 0) {
            Throw string("Division by error");
        }
    } else {
        cout << a/b;
    }
}
catch ( string e ) {
    cout << e << endl;
}
return 0;

```

Note: If you give in normal string, it will send as string

How works:

1. code inside try runs.
2. if error occurs → Throw
3. control immediately jumps to catch
4. Remaining try block is skipped.

* multiple catch box:

```
try {  
    Throw 10;  
}  
catch (int e) {  
    cout << "Integer error" << endl;  
}  
catch (double e) {  
    cout << "float or double error" << endl;  
}
```

* Catch all Exceptions.

```
catch (...) {  
    cout << "unknown error occurred!" ;  
}
```

* Standard Exceptions in C++

(#include <exception>)

Standard exception library

-> common ones:

std::exception

std::runtime_error

std::out_of_range

std::invalid_argument

std::bad_alloc

(Learn it later - don't forget) ✕

