OBJECT:

- Simple data types, variables and constants,
- Arithmetic operators, arithmetic assignment operators, increment and decrement operators
- Escape sequences.

RULES FOR NAMING VARIABLES:

- Variable names must begin with a letter or names can also begin with $ and _
- Variable names can contain letters, numbers, the underscore (_) and the doller ($) sign.
- Variable names cannot contain spaces.
- Variable name cannot be a keyword or a reserved word.
- Variable names cannot contain special characters such as all operators, #, <, +, etc.
- C++ is a case-sensitive language. Variable names written in capital letters differ from variable names with the same name but written in small letters. For example, the variable name EXFORSYS differs from the variable name exforsys.

DATA TYPES

You may like to store information of various data types like character, wide character, integer, floating point, double floating point, boolean etc. Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory.

Data Type	Range	No. of Bits in Memory
Integers: Used to store whole numbers. C++ supports several integer types, having different sizes for storing signed (both negative and positive) and unsigned numbers		
short	-32768 → 32767	16 bit
unsigned short	0 → 65535	16 bit
Int	-32768 → 32767 -2147483648 → 2147483647	16 bit or 32 bit
unsigned int	0 → 65535 0 → 4294967295	16 bit or 32 bit
Long	-2147483648 → 2147483647	32 bit
unsigned long	0 → 4294967295	32 bit
long long	-9223372036854775808 → 9223372036854775807	64 bit
unsigned long long	0 → 18446744073709551615	64 bit
Floating point: Floating point types are used to store numbers that contain decimal point (real numbers). C++ supports several floating point types, having different sizes.		
float	10^{-38} → 10^{38}	32 bit
double	10^{-308} → 10^{308}	64 bit
long double	10^{-308} → 10^{308}	64 bit
Character: Used to hold single character can also store number (ASCII Code of a character)		
Char	Can also store integers (-128 → 127)	8 bit
unsigned char	Can also store integers (0 → 255)	8 bit
Boolean: Used two store only two types of values TRUE or FALSE. Can be used in decision making statements.		
bool	True / False	8 bit

DECLARING VARIABLES

Declaring means giving a name and a data type to a variable before it can be used. Syntax:

```
datatype variable_name;
datatype variable_name1, variable_name2, ..;
```

THE ASSIGNMENT OPERATOR AND INITIAL VALUES

When you declare a variable, you can also assign first value to the databy using the assignment operator (=)

```
datatype variableName = initialValue;
datatype var1=initialValue1, var2=initialValue2 ;
```

Notice that assignment is right to left. The initial value is assigned to the variable.

INTEGER VALUES

Can be expressed in decimal(base 10)
Hexadecimal(base 16) : Prefix 0 is used to indicate octal
Octal(base 8) : prefix 0x indicates hexadecimal
Examples:

```
short decimal = 100;   OR   int decimal = -230;
int octal = 0144;      OR   long octal = -0234;
int hexa = 0x4F;       OR   int hexa = -0x2A;

unsigned short decimal = 100;
unsigned int octal = 0144;
unsigned long hexa = 0x4F;
```

FLOATING-POINT VALUES

The floating point types (float and double) can also be expressed using E or e (for scientific notation),
Examples:

```
double d1 = 123.4;
same value as d1, but in scientific notation
double d2 = 1.234e2;       ←   1.234x10^2
float f1  = -123.4;
float f1  = 1.7E-3;        ←   1.7x10^-3
```

CHAR VALUES

Character values can be given by using single quote :

```
char ch = ‘N';
```

Character can also have integral values: Unicode value in Decimal, Octal and Hexadecimal.
Examples:

```
char ch = 78;            //range is 0 to 255
char ch = 0116;         //range 000 to 377
char ch = '\u4E';       // range \u00 to \uFF
```

Character values can also be escape sequence be given by using single quote :

```
char ch = '\n';
```

PROGRAM 1:

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

int main()  {

short s = 1234;              //Short Integer type variable
unsigned short us = 41234;  //unsigned Short

int i = 3212;                //Integer type variable
unsigned int ui = 54321;    //unsigned int

long l = 123345435;              //Long Integer type variable
unsigned long ul = 4111117295;  //unsigned Long

long long ll = -7612334543534454;            //Long Long
unsigned long long ull = 9451111172955545;  //Unsigned Long Long

float f = 123.432;           //Floating point type variable
double d = 4.12e223;         //Double precision float type

char ch1 = 'N';              //Character type variable
unsigned char ch2 = 203;    //unsigned character

bool q = true;     //Boolean Type Variable

//Printing all variable values on screen

cout << "short variable= " << s << endl;
cout << "unsigned short variable= " << us << endl;
cout << "int variable= " << i << endl;
cout << "unsigned int variable= " << ui <<endl;
cout << "long variable= " << l << endl;
cout << "unsigned long variable= " << ul <<endl;
cout << "long long variable= " << ll << endl;
cout << "unsigned long long variable= " << ull <<endl;
cout << "float variable= " << f << endl;
cout << "double variable= " << d << endl;
cout << "char variable= " << ch1 << endl;
cout << "unsigned char variable= " << ch2 << endl;
cout << "bool variable= " << q << endl;

return 0;
}
```

ESCAPE SEQUENCES SUMMARY:

Escape Sequence	Description
\'	Single quote
\”	Double quote
\\	Backslash
\nnn	Octal number (nnn)
\0	Null character (really just the octal number zero)
\a	Audible bell
\b	Backspace
\f	Formfeed
\n	Newline
\r	Carriage return
\t	Horizontal tab
\xnnn	Hexadecimal number (nnn)

PROGRAM 2:

```
#include <iostream>
using namespace std;
int main() {

char ch1 = 'A';    //define char variable as character
char ch2 = '\t';   //define char variable as tab

cout << ch1 << ch2;       //display characters

ch1 = 'B';         //set char variable to char constant

cout << ch1;       //display character
cout << '\n';           //display newline character

return 0;
}
```

CONSTANTS:

Constants, like variables, contain data type but constants have fixed value and that value cannot be changed in the program. There are 2 ways to declare a constant

- const data_type const_name = value; (Example: const int PI = 3.14;)
- #define const_name value (Example: #define PI 3.14)

PROGRAM 3:

```
#include <iostream>
using namespace std;
int main(){
float rad;                                  //variable of type float
const float PI = 3.14159;                  //type const float
cout << "Enter radius of circle: ";  //prompt
cin >> rad;                                 //get radius
float area = PI * rad * rad;         //find area
cout << "Area is " << area << endl;  //display answer
return 0;
}
```

REMAINDER (MODULUS) OPERATOR

Take a simple arithmetic problem: what's left over when you divide 11 by 3? The answer is easy to compute: divide 11 by 3 and take the remainder: 2. The modulus operator ('%'), that computes the remainder that results from performing **integer** division.

PROGRAM 4:

```
#include <iostream>
using namespace std;
int main() {
cout <<  6 % 8 << endl    // 6
     <<  8 % 8 << endl    // 0
     <<  9 % 8 << endl    // 1
     << 10 % 8 << endl;   // 2
return 0;
}
```

INCREMENT AND DECREMENT OPERATORS

The increment operator ++ adds 1 to its operand, and the decrement operator -- subtracts 1 from its operand.

`x = x+1;` is the same as `x++;`

And

`x = x-1;` is the same as `x--;`

Both the increment and decrement operators can either precede (prefix) or follow (postfix) the operand.
For example:

`x = x+1;` can be written as

`++x; // prefix form`

or as

`x++; // postfix form`

When an increment or decrement is used as part of an expression, there is an important difference in prefix and postfix forms. If you are using prefix form then increment or decrement will be done before rest of the expression, and if you are using postfix form, then increment or decrement will be done after the complete expression is evaluated.

PROGRAM 5:

```
#include <iostream>
using namespace std;
int main() {
int c = 10;
cout << "c =" << c << endl;    //displays 10
cout << "++c =" << ++c << endl;  //displays 11 (prefix)
cout << "c =" << c << endl;    //displays 11
cout << "c++ =" << c++ << endl;  //displays 11 (postfix)
cout << "c =" << c << endl;    //displays 12
cout << "--c =" << --c << endl;  //displays 11 (prefix)
cout << "c =" << c << endl;    //displays 11
cout << "c-- =" << c-- << endl;  //displays 11 (postfix)
cout << "c =" << c << endl;    //displays 10
return 0;
}
```

ARITHMETIC ASSIGNMENT OPERATORS:

+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand.	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand.	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand.	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand.	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand.	C %= A is equivalent to C = C % A

PROGRAM 6:

```
#include <iostream>
using namespace std;
int main() {
int ans = 27;
ans += 10;            //same as: ans = ans + 10;
cout << ans << ", ";
ans -= 7;             //same as: ans = ans - 7;
cout << ans << ", ";
ans *= 2;             //same as: ans = ans * 2;
cout << ans << ", ";
ans /= 3;             //same as: ans = ans / 3;
cout << ans << ", ";
ans %= 3;             //same as: ans = ans % 3;
cout << ans;
return 0;
}
```

EXERCISE 1:

Write a program which generate the following output on the screen you have to use scape sequences and you cannot use endl. Use one or more than one cout statements to generate output.

```
“Welcome All” is a string constant
‘A’ is a character constant
\t is for tab
\n for new line
```

EXERCISE 2:

Write a program which takes 2 **int** values from user and then perform all five arithmetic operations on them. The output should be like:

```
Enter First Number: 12
Enter Second Number: 32
12 + 32 = 44
12 - 32 = -20
12 x 32 = 384
12 / 32 = 0.375
12 % 32 = 11
```

User can give different values and output will be different every time. You will observe that the division operator is not giving the correct answer with ‘int’ data type.

Change the variable data type from **int** to **float** then observe if there is any other problem or error.

We will correct it in week 4. Also, you will notice if you take float variable than remainder operator will not work.

EXERCISE 1:

Write a program in C++ to calculate the volume of a sphere. Use the formula:

$$V = \frac{4}{3}\pi r^3$$

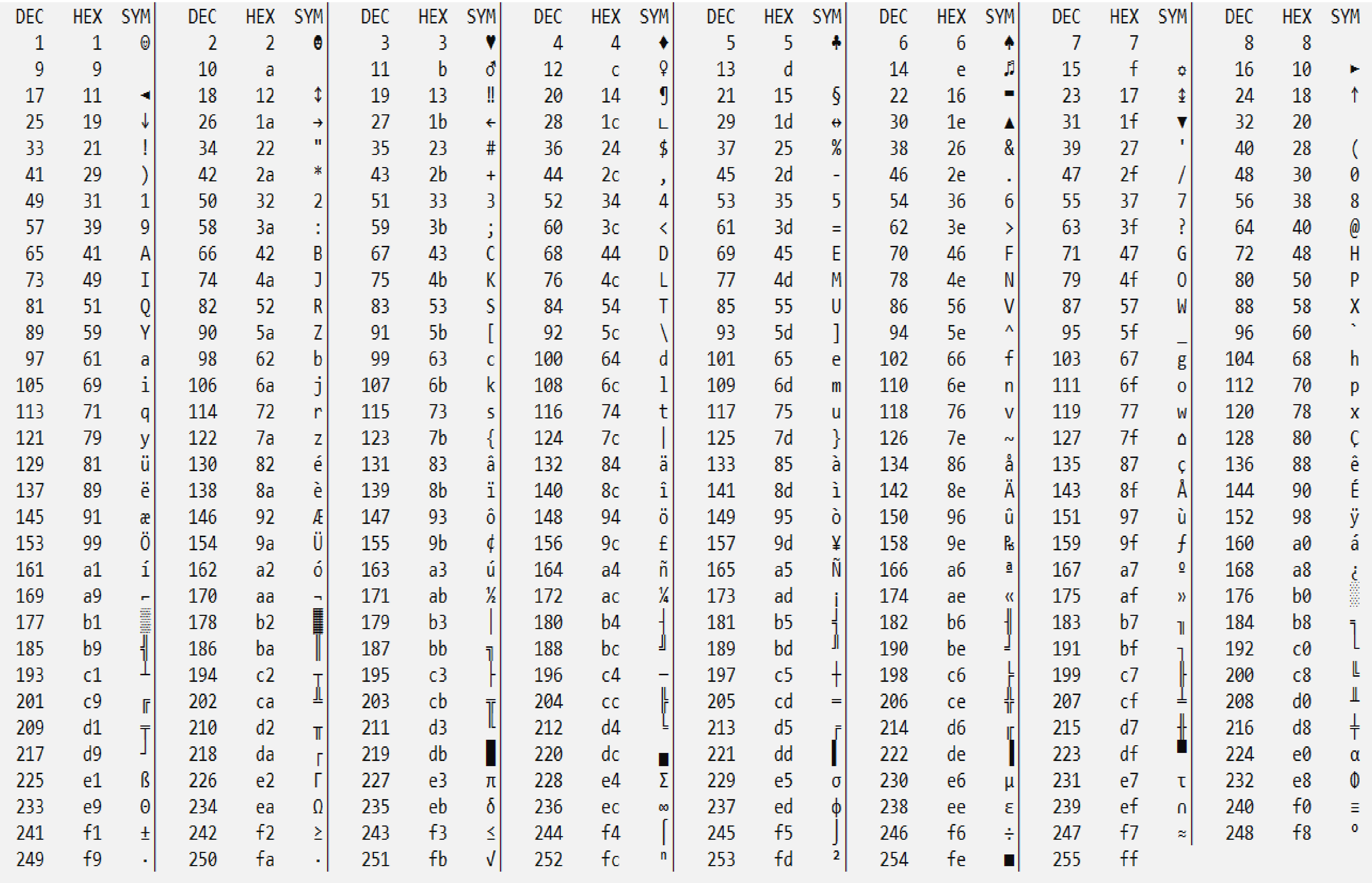
The Volume and Radius variables should be of **float** datatypes.

Create a constant **PI** for the value of PI and use it.

Your output should be:

```
Input the radius of a sphere : 6
The volume of a sphere is : 904.32
```

Observe the output is correct or not than to correct it use 4.0/3 instead of 4/3. The reason will be discussed in next week.



DEC	HEX	SYM	DEC	HEX	SYM	DEC	HEX	SYM	DEC	HEX	SYM	DEC	HEX	SYM	DEC	HEX	SYM	DEC	HEX	SYM	DEC	HEX	SYM
1	1	☺	2	2	☻	3	3	♥	4	4	♦	5	5	♣	6	6	♠	7	7		8	8	
9	9		10	a		11	b	♂	12	c	♀	13	d		14	e	♫	15	f	☼	16	10	►
17	11	◄	18	12	↕	19	13	‼	20	14	¶	21	15	§	22	16	▬	23	17	↨	24	18	↑
25	19	↓	26	1a	→	27	1b	←	28	1c	∟	29	1d	↔	30	1e	▲	31	1f	▼	32	20	
33	21	!	34	22	"	35	23	#	36	24	$	37	25	%	38	26	&	39	27	'	40	28	(
41	29	)	42	2a	*	43	2b	+	44	2c	,	45	2d	-	46	2e	.	47	2f	/	48	30	0
49	31	1	50	32	2	51	33	3	52	34	4	53	35	5	54	36	6	55	37	7	56	38	8
57	39	9	58	3a	:	59	3b	;	60	3c	<	61	3d	=	62	3e	>	63	3f	?	64	40	@
65	41	A	66	42	B	67	43	C	68	44	D	69	45	E	70	46	F	71	47	G	72	48	H
73	49	I	74	4a	J	75	4b	K	76	4c	L	77	4d	M	78	4e	N	79	4f	O	80	50	P
81	51	Q	82	52	R	83	53	S	84	54	T	85	55	U	86	56	V	87	57	W	88	58	X
89	59	Y	90	5a	Z	91	5b	[	92	5c	\	93	5d	]	94	5e	^	95	5f	_	96	60	`
97	61	a	98	62	b	99	63	c	100	64	d	101	65	e	102	66	f	103	67	g	104	68	h
105	69	i	106	6a	j	107	6b	k	108	6c	l	109	6d	m	110	6e	n	111	6f	o	112	70	p
113	71	q	114	72	r	115	73	s	116	74	t	117	75	u	118	76	v	119	77	w	120	78	x
121	79	y	122	7a	z	123	7b	{	124	7c	|	125	7d	}	126	7e	~	127	7f	⌂	128	80	Ç
129	81	ü	130	82	é	131	83	â	132	84	ä	133	85	à	134	86	å	135	87	ç	136	88	ê
137	89	ë	138	8a	è	139	8b	ï	140	8c	î	141	8d	ì	142	8e	Ä	143	8f	Å	144	90	É
145	91	æ	146	92	Æ	147	93	ô	148	94	ö	149	95	ò	150	96	û	151	97	ù	152	98	ÿ
153	99	Ö	154	9a	Ü	155	9b	¢	156	9c	£	157	9d	¥	158	9e	₧	159	9f	ƒ	160	a0	á
161	a1	í	162	a2	ó	163	a3	ú	164	a4	ñ	165	a5	Ñ	166	a6	ª	167	a7	º	168	a8	¿
169	a9	⌐	170	aa	¬	171	ab	½	172	ac	¼	173	ad	¡	174	ae	«	175	af	»	176	b0	░
177	b1	▒	178	b2	▓	179	b3	│	180	b4	┤	181	b5	╡	182	b6	╢	183	b7	╖	184	b8	╕
185	b9	╣	186	ba	║	187	bb	╗	188	bc	╝	189	bd	╜	190	be	╛	191	bf	┐	192	c0	└
193	c1	┴	194	c2	┬	195	c3	├	196	c4	─	197	c5	┼	198	c6	╞	199	c7	╟	200	c8	╚
201	c9	╔	202	ca	╩	203	cb	╦	204	cc	╠	205	cd	═	206	ce	╬	207	cf	╧	208	d0	╨
209	d1	╤	210	d2	╥	211	d3	╙	212	d4	╘	213	d5	╒	214	d6	╓	215	d7	╫	216	d8	╪
217	d9	┘	218	da	┌	219	db	█	220	dc	▄	221	dd	▌	222	de	▐	223	df	▀	224	e0	α
225	e1	ß	226	e2	Γ	227	e3	π	228	e4	Σ	229	e5	σ	230	e6	µ	231	e7	τ	232	e8	Φ
233	e9	Θ	234	ea	Ω	235	eb	δ	236	ec	∞	237	ed	φ	238	ee	ε	239	ef	∩	240	f0	≡
241	f1	±	242	f2	≥	243	f3	≤	244	f4	⌠	245	f5	⌡	246	f6	÷	247	f7	≈	248	f8	°
249	f9	∙	250	fa	·	251	fb	√	252	fc	ⁿ	253	fd	²	254	fe	■	255	ff				