Home Work No.2

PF (FALL-2022)

8th September 2022

Content:

1) C++ operators
2) Practices codes set precision an set width
3) Operators
4) Practice Codes of operators
5) Read Chapter 2 and 3
6) ASCII table
7) Read Chapter 2 and 3 (3.7 for iomanip Page 138 to 148)

Read homework carefully from start until End. You can attempt it in multiple times. You must read Topic 3.7 for iomanip Page 138 to 148 before attempting this homework.

Some important things to know from last three Lectures:

C++ is/has:

1. Strictly typed
2. Static typed
3. Compiled language
4. Faster and resource and performance efficient code.
5. Designed to develop Operating system and embedded software
6. Curly brackets { … } used for blocks
7. Block have local scope
8. Collection of statements/Commands
9. Statement terminator ";"
10. Contains most of programming constructs
11. Best programming language for teaching and learning?
12. Case sensitive language

Remember:

- Data is stored in binary form in system memory (Von Neumann Architecture).
- By default data is signed (negative and positive)
- Representation of binary data is in the form of 2's complement
- Integral data [char (1byte), short (2 bytes), int (4 bytes) and long (8 bytes)] are by default signed and stored in 2's complement binary form.
- We can make Integral data unsigned explicitly by using reserved keyword unsigned.
- Floating-point data float (4-bytes), double (8-bytes) and long double (16-bytes) are represented in binary notation using IEEE-32 bit and IEEE-64 bit floating point binary representation.
- Address of a variable does not depend upon type of a variable. It depends upon the architecture of System either 4 byte (32 bit) or 8 byte (64-bit).
- Operator precedence vs operator associativity
- Operator Arity.

Run following programs in separate .cpp files and carefully study and understand it.

Code Explained in class Lecture.

……code.cpp……..

```
#include <iostream>
#include<cstdlib>
#include<time.h>
#include<fstream>

using namespace std;

int main()
{
	int integer1;  // first number to be input by user
	int integer2;  // second number to be input by user
	int integer3; //Third number to be input by user
	int sum;       // variable in which sum will be stored
	int Average;   // variable in which Average will be stored

	/*cout << "Enter first integer\n";  // prompt for input
	cin >> integer1;                  // read an integer

	cout << "Enter second integer\n"; // prompt for input
	cin >> integer2;                  // read an integer

	cout << "Enter Third integer\n"; // prompt for input
	cin >> integer3;                  // read an integer

	sum = integer1 + integer2 + integer3;  // assign result to sum

	cout << "\nSum is :" << sum <<endl; // print sum

	Average = sum / 3;

	cout << "\nAverage is :" << Average; //Print Average;


	return 0;   // indicate that program ended successfully*/

	cout << "Enter three integers\n";  // prompt for three inputs
	cin >> integer1>>integer2>>integer3;                  // read an integer


	sum = integer1 + integer2 + integer3;  // assign result to sum

	cout << "\nSum is :" << sum << endl; // print sum

	Average = sum / 3;

	cout << "\nAverage is :" << Average; //Print Average;


	return 0;   // indicate that program ended successfully
```

```
}
```

Notice carefully the difference between 1.cpp and 2.cpp.

```
//……1.cpp…….
#include <iostream>
#include<iomanip>
using namespace std;
int main ()
{
	float f = 12.5544;
	cout<<setprecision(6)<<f<<endl;
	cout<<setprecision(5)<<f<<endl;
	cout<<setprecision(4)<<f<<endl;
	cout<<setprecision(3)<<f<<endl;
	cout<<setprecision(2)<<f<<endl;
	cout<<setprecision(1)<<f<<endl;
	cout<<endl;
	return 0;
}
//……2.cpp……
#include <iostream>
#include<iomanip>
using namespace std;
int main ()
{
	float f = 12.5544;
	cout<<setprecision(6)<<fixed<<f<<endl;
	cout<<setprecision(5)<<fixed<<f<<endl;
	cout<<setprecision(4)<<fixed<<f<<endl;
```

```
        cout<<setprecision(3)<<fixed<<f<<endl;
        cout<<setprecision(2)<<fixed<<f<<endl;
        cout<<setprecision(1)<<fixed<<f<<endl;
        cout<<endl;
        return 0;
}
```

//……..3.cpp…….

/* This program will explain how to print output using setprecision and setw. This program asks for sales figures for 3 days. The total sales are calculated and displayed in a table.*/

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
 {
         double day1, day2, day3, total;

        // Get the sales for each day.
        cout << "Enter the sales for day 1: ";
        cin >> day1;
        cout << "Enter the sales for day 2: ";
        cin >> day2;
        cout << "Enter the sales for day 3: ";
        cin >> day3;
        // Calculate the total sales.
        total = day1 + day2 + day3;
        // Display the sales figures.
        cout << "\nSales Figures\n";
```

```
        cout << "-------------\n";
        cout << setprecision(2) << fixed;
        cout << "Day 1: " << setw(8) << day1 << endl;
        cout << "Day 2: " << setw(8) << day2 << endl;
        cout << "Day 3: " << setw(8) << day3 << endl;
        cout << "Total: " << setw(8) << total << endl;
        return 0;
}
///……..4.cpp……..
/* This program explains how to print Pattern using setfill() and setw. */
#include <iostream>
#include<iomanip>
using namespace std;
int main() {
        std::cout  << std::setw(10) << std::setfill('x');
        std::cout << '#' << std::endl;
        std::cout << std::setw(8) << std::setfill('x');
        std::cout << '#' << std::endl;
        std::cout << std::setw(6) << std::setfill('x');
        std::cout << '#' << std::endl;
        std::cout << std::setw(4) << std::setfill('x');
        std::cout << '#' << std::endl;
        std::cout << std::setw(2) << std::setfill('x');
        std::cout << '#' << std::endl;
        std::cout << '#' << std::endl;

}
```

……..operators……

Common C++ operators:

assignment	*increment decrement*	*arithmetic*	*logical*	*comparison*	*member access*	*other*

`a = b`
`a += b`
`a -= b`
`a *= b`
`a /= b`
`a %= b`
`a &= b`
`a |= b`
`a ^= b`
`a <<= b`
`a >>= b`
	`++a`
	`--a`
	`a++`
	`a--`
		`+a`
		`-a`
		`a + b`
		`a - b`
		`a * b`
		`a / b`
		`a % b`
		`~a`
		`a & b`
		`a | b`
		`a ^ b`
		`a << b`
		`a >> b`
			`!a`
			`a && b`
			`a || b`
				`a == b`
				`a != b`
				`a < b`
				`a > b`
				`a <= b`
				`a >= b`
				`a <=> b`
					`a[b]`
					`*a`
					`&a`
					`a->b`
					`a.b`
					`a->*b`
					`a.*b`
						`a(...)`
						`a, b`
						`? :`

Special operators						
`static_cast` converts one type to another related type **`dynamic_cast`** converts within inheritance hierarchies **`const_cast`** adds or removes cv qualifiers **`reinterpret_cast`** converts type to unrelated type C-style cast converts one type to another by a mix of **`static_cast`**, **`const_cast`**, and **`reinterpret_cast`** **`new`** creates objects with dynamic storage duration **`delete`** destructs objects previously created by the new expression and releases obtained memory area **`sizeof`** queries the size of a type **`sizeof...`** queries the size of a parameter pack (since C++11) **`typeid`** queries the type information of a type **`noexcept`** checks if an expression can throw an exception (since C++11) **`alignof`** queries alignment requirements of a type (since C++11)						

List of C/C++ operators their Precedence and Associativity:

Precedence	Operator	Description	Associativity
1	**: :**	**Scope resolution**	

2	`a++ a--`	**Suffix/postfix increment and decrement**	Left-to-right
	`type``()` *`type`*`{}`	**Functional cast**	
	`a()`	**Function call**	
	`a[]`	**Subscript**	
	`. ->`	**Member access**	
3	`++a --a`	**Prefix increment and decrement**	Right-to-left
	`+a -a`	**Unary plus and minus**	
	`! ~`	**Logical NOT and bitwise NOT**	
	`(`*`type`*`)`	**C-style cast**	
	`*a`	**Indirection (dereference)**	
	`&a`	**Address-of**	
	`Sizeof`	**Size-of**	
	`new new[]`	**Dynamic memory allocation**	
	`delete delete[]`	**Dynamic memory deallocation**	
4	`.* ->*`	**Pointer-to-member**	

5	`a*b a/b a%b`	**Multiplication, division, and remainder**	Left-to-right
6	`a+b a-b`	**Addition and subtraction**	
7	`<< >>`	**Bitwise left shift and right shift**	
8	`<=>`	**Three-way comparison operator (since C++20)**	
9	`< <=` `> >=`	**For relational operators < and ≤ respectively** **For relational operators > and ≥ respectively**	
10	`== !=`	**For relational operators = and ≠ respectively**	
11	`&`	**Bitwise AND**	
12	`^`	**Bitwise XOR (exclusive or)**	
13	`|`	**Bitwise OR (inclusive or)**	
14	`&&`	**Logical AND**	
15	`||`	**Logical OR**	

16	`a?b:c`
	`Throw`
	`=`
		Ternary conditional
		throw operator
		Direct assignment (provided by default for C++ classes)	

	`+=` `-=`	**Compound assignment by sum and difference**	Right-to-left
	`*=` `/=` `%=`	**Compound assignment by product, quotient, and remainder**	
	`<<=` `>>=`	**Compound assignment by bitwise left shift and right shift**	
	`&=` `^=` `|=`	**Compound assignment by bitwise AND, XOR, and OR**	
17	**,**	**Comma**	Left-to-right

What will be the outputs of following expressions?

First, solve by hand and then program these expressions and see output on screen?

1. **1777 / 5 % 36 / 13**
2. **10 / 2 - 3**
3. **8 + 12 * 2 - 4**
4. **4 + 17 % 2 - 1**
5. **6 - 3 * 2 + 7 – 1**
6. **28 / 4 - 2**
7. **6 + 12 * 2 - 8**
8. **4 + 8 * 2**
9. **(6 + 12) * 2 - 8**
10. **6 + 17 % 3 - 2**
11. **2 + 22 * (9 - 7)**
12. **(8 + 7) * 2**
13. **(16 + 7) % 2 - 1**
14. **12 / (10 - 6)**
15. **(19 - 3) * (2 + 2) / 4**
16. **!((7 / 12 < 15) || (8 * 0 && 10)) + (19.5 < 30 - 10)**
17. **7 + 7 == 70 / 5 * 1 + 25 % 5**
18. **(5 > 7) * 10 + 5 * 2 / 2 < 5**
19. **14 / (11 - 4)**

20. **9 + 12 * (8 - 3)**
21. **(6 + 17) % 2 – 1**
22. **(9 - 3) * (6 + 9) / 3**
23. **~(3 & 3 & 14)**

…… Code Dry Run……

Dry run the following codes:

1)

```
#include <iostream>
#include<iomanip>
using namespace std;
int main() {
        cout << setprecision(6);
        double d = 33.00;
        int x = 15;
        cout << d / x << endl;
        cout << setprecision(6) << fixed;
        cout << d / x << endl;
        return 0;
}
```

2)

```
#include <iostream>
#include<iomanip>
using namespace std;
int main() {
        cout << setprecision(4);
        double d = 3.00;
        int x = 12;
        cout << d / x << endl;
```

```
                cout << setprecision(4) << fixed;
                cout << d / x << endl;
                return 0;
}
```

3)

```
#include <iostream>
#include<iomanip>
using namespace std;
int main() {
        int a=26, b=3;
        a = a / b;
        b = b - a;
        b = b / 2;
        a = a * b;
        cout << "\n First Modified value : " << a;
        cout << "\n Second Modified value: " << b;
}
```

4)

```
#include <iostream>
#include<iomanip>
using namespace std;
int main() {
        int a=3, b=26;
        a = a / b;
        b = b - a;
        b = b / 2;
```

```
        a = a * b;
        cout << "\n First Modified value : " << a;
        cout << "\n Second Modified value: " << b;
}
```

5)

```
#include <iostream>
#include<iomanip>
using namespace std;
int main() {
        int z = 5, j = 7, k = 6, n = 3;
        cout << z + j % k + k * n - 15 << endl;
        cout << z % n + 5 << endl;
}
```

6)

```
#include <iostream>
#include<iomanip>
using namespace std;
int main()
{
        int i, j , k = 3;
        i -= j -= k;
        cout << i << j << k;
        return 0;
}
```

7)

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

```
int main() {
	unsigned short j = -2;
	cout <<"\nValue of \"j\" is = " << j << endl;
	return 0;
}
```

8)

```
#include <iostream>
#include<iomanip>
using namespace std;
int main() {
	short i = 32769;
	unsigned short j = 32769;
	cout <<"\nValue of \"i\" is = "<< i <<endl;
	cout <<"\nValue of \"j\" is = " << j << endl;
return 0;
}
```

Run following programs in separate .cpp files and carefully study and understand it.

```
//…… 1.cpp……
#include <iostream>
#include<iomanip>
using namespace std;
int main() {
	const double I;
	int n;
	I = 3.14159265358979;
```

```
	cout << I * I;
}
//…… 2a.cpp……
#include <iostream>
using namespace std;
int main ()
{
	int a=10;
	short b=10;
	long c=10;
	cout<<"value of Integer a is  "<<a<<"  size is "<<sizeof(a)<<endl;
	cout<<"value of short c is  "<<b<<"  size is "<<sizeof(b)<<endl;
	cout<<"value of long c is  "<<c<<"  size is "<<sizeof(c)<<endl;
	cout<<"size of integer Literal is 10 and its size is "<<sizeof(10)<<endl;
	return 0;
}
```

Dry run the following code and then run it on compiler.

```
//…… 2b.cpp……
#include <iostream>
using namespace std;
int main ()
{
	float a = 10.01;
	double b = 10.01;
	long double c = 10.01;
	cout << "value of Integer a is  " << a << "  size is " << sizeof(a) << endl;
```

```
	cout << "value of short c is  " << b << "  size is " << sizeof(b) << endl;
	cout << "value of long c is  " << c << "  size is " << sizeof(c) << endl;
	cout << "size of integer Literal is 10 and its size is " << sizeof(10.01) << endl;
	return 0;
 }
//…… 3.cpp……
#include <iostream>
#include<iomanip>
using namespace std;
int main() {
	int x = 4;
	int y = 6;
	double z = 7;
	int num2 = z / y * x / 2 * 10 + (y * x + 2) / z;
	cout << "\n\tThe value of expression 1 is " << num2 << "." << endl;
	double num3 = z * x + y * z / 16;
	cout << "\n\tThe value of expression 2 is " << num3 << "." << endl;
}
```

Run following programs in separate .cpp files and carefully study and understand it.

```
//……..1.cpp……
#include <iostream>
#include<iomanip>
using namespace std;
int main ()
{
	cout<<endl;
	cout<<setw(20)<<left<<"Hello";
```

```
	cout<<setw(20)<<right<<"bye";
	cout<<endl;
	return 0;
}
//……..2.cpp……
#include <iostream>
#include<iomanip>
using namespace std;
int main() {
	cout << endl;
	cout << setw(20) << left << "Hello";
	cout << endl;
	cout << setw(20) << right << "bye";
	cout << endl;

}
```

Challenge question1: Run the following code and understand it carefully.

```
///……..4.cpp……..

#include <iostream>
using namespace std;
int main()
{
	short cheeta;//declared a short variable alpha
	cheeta = 100;//assigned alpha variable with decimal value
	short alpha;//declared a short variable alpha
	alpha = 0b0000100;//assigned alpha variable with binary value
	short beta;//declared an short variable beta
	beta = 0100;//assigned alpha variable with octal value
```

```
	short gama;//declared an short variable gama
	gama = 0x100;//assigned alpha variable with octal value

	cout << "\nValue of variable \"cheeta\" is \t  " << cheeta;
	cout << "\nValue of variable \"alpha\" is \t  " << alpha;
	cout << "\nValue of variable \"beta\" is \t  " << beta;
	cout << "\nValue of variable \"gama\" is \t  " << gama;
	return 0;

}
```

Challenge question2: Run the following code and understand it carefully.

```
///……..5.cpp……..
/* This program explains how to print Pattern using setfill() and setw. */
#include <iostream>
#include<iomanip>
using namespace std;
int main() {
	std::cout  << std::setw(10) << std::setfill('x');
	std::cout << '#' << std::endl;
	std::cout << std::setw(8) << std::setfill('x');
	std::cout << '#' << std::endl;
	std::cout << std::setw(6) << std::setfill('x');
	std::cout << '#' << std::endl;
	std::cout << std::setw(4) << std::setfill('x');
	std::cout << '#' << std::endl;
	std::cout << std::setw(2) << std::setfill('x');
	std::cout << '#' << std::endl;
	std::cout << '#' << std::endl;
```

}

Challenge question: Find the errors in the following code Segment and correct this code?

```
int main()
{
        float 5f=1.1
        int a = 15;
        Cout<<"Value for a is">>a;
        cout>>"\tSize of a is"<<sizeof(a;
        Cout<<\n"Value for f is"<<5F;
        cout>>\t"Size of f is"<<Sizeof(5f);
        int a =15;
        Cout<<"\n Now for a is"<<a;
        cout>>"\tSize of a is<<sizeof(a);

}
```

Ascii	Char	Ascii	Char	Ascii	Char	Ascii	Char
0	Null	32	Space	64	@	96	`
1	Start of heading	33	!	65	A	97	a
2	Start of text	34	"	66	B	98	b
3	End of text	35	#	67	C	99	c
4	End of transmit	36	$	68	D	100	d
5	Enquiry	37	%	69	E	101	e
6	Acknowledge	38	&	70	F	102	f
7	Audible bell	39	'	71	G	103	g
8	Backspace	40	(	72	H	104	h
9	Horizontal tab	41	)	73	I	105	i
10	Line feed	42	*	74	J	106	j
11	Vertical tab	43	+	75	K	107	k
12	Form feed	44	,	76	L	108	l
13	Carriage return	45	-	77	M	109	m
14	Shift in	46	.	78	N	110	n
15	Shift out	47	/	79	O	111	o
16	Data link escape	48	0	80	P	112	p
17	Device control 1	49	1	81	Q	113	q
18	Device control 2	50	2	82	R	114	r
19	Device control 3	51	3	83	S	115	s
20	Device control 4	52	4	84	T	116	t
21	Neg. acknowledge	53	5	85	U	117	u
22	Synchronous idle	54	6	86	V	118	v
23	End trans. block	55	7	87	W	119	w
24	Cancel	56	8	88	X	120	x
25	End of medium	57	9	89	Y	121	y
26	Substitution	58	:	90	Z	122	z
27	Escape	59	;	91	[	123	{
28	File separator	60	<	92	\	124	|
29	Group separator	61	=	93	]	125	}
30	Record separator	62	>	94	^	126	~
31	Unit separator	63	?	95	_	127	Forward del.

Ascii	Char	Ascii	Char	Ascii	Char	Ascii	Char
128	Ä	160	†	192	¿	224	‡
129	Å	161	°	193	¡	225	·
130	Ç	162	¢	194	¬	226	‚
131	É	163	£	195	√	227	„
132	Ñ	164	§	196	ƒ	228	‰
133	Ö	165	•	197	≈	229	Â
134	Ü	166	¶	198	∆	230	Ê
135	á	167	ß	199	«	231	Á
136	à	168	®	200	»	232	Ë
137	â	169	©	201	…	233	È
138	ä	170	™	202		234	Í
139	ã	171	´	203	À	235	Î
140	å	172	¨	204	Ã	236	Ï
141	ç	173	≠	205	Õ	237	Ì
142	é	174	Æ	206	Œ	238	Ó
143	è	175	Ø	207	œ	239	Ô
144	ê	176	∞	208	–	240	
145	ë	177	±	209	—	241	Ò
146	í	178	≤	210	“	242	Ú
147	ì	179	≥	211	”	243	Û
148	î	180	¥	212	‘	244	Ù
149	ï	181	µ	213	’	245	ı
150	ñ	182	∂	214	÷	246	ˆ
151	ó	183	∑	215	◊	247	˜
152	ò	184	∏	216	ÿ	248	¯
153	ô	185	π	217	Ÿ	249	˘
154	ö	186	∫	218	⁄	250	˙
155	õ	187	ª	219	€	251	˚
156	ú	188	º	220	‹	252	¸
157	ù	189	Ω	221	›	253	˝
158	û	190	æ	222	fi	254	˛
159	ü	191	ø	223	fl	255	ˇ