

DASL-100.2

C++ Programming and Linux

Week 2-1

1. Data types.
2. Data input and output.
3. Operators.
4. Decision Making.
5. Loops.
6. Debugging.

DASL-100.2

C++ Programming and Linux

1. Data Type

- In C++, data types are used to define the type of data that a variable will store. The following are some of the most commonly used data types in C++:

	Definition	*Memory (32-bit sys)	Value range
int	Store integers (whole number)	4 bytes	-2147483648 to 2147483647
float	Store floating-point number(decimal number)	4 bytes	-3.4 x 10 ³⁸ to 3.4 x 10 ³⁸ (~7 decimal digits precision)
double	Store double-precision floating-point numbers (numbers with a larger range and precision than float)	8 bytes	-1.7 x 10 ³⁰⁸ to 1.7 x 10 ³⁰⁸ (~15 decimal digits precision)
char	Store individual characters	1 byte	0 to 255 (or -128 to 127)
bool	Store boolean values (true or false)	1 byte	0 or 1 (true or false)

DASL-100.2

C++ Programming and Linux

1. Data Type

- *Memory space occupied by each data type depends on the compiler and system architecture.
- On a 64-bit system, the memory space occupied by each data type may be different, and larger data types may occupy more memory.
- 1 bit can store 0 and 1.
- 1 byte = 8 bit. Example:

BIT#:	7	6	5	4	3	2	1	0
	0	0	1	0	1	1	0	1
VALUE:	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1

Bits and Bytes Ref. Wikipedia

Binary	Oct	Dec	Hex	Glyph	Binary	Oct	Dec	Hex	Glyph	Binary	Oct	Dec	Hex	Glyph
010 0000	040	32	20	space	100 0000	100	64	40	@	110 0000	140	96	60	.
010 0001	041	33	21	!	100 0001	101	65	41	A	110 0001	141	97	61	a
010 0010	042	34	22	"	100 0010	102	66	42	B	110 0010	142	98	62	b
010 0011	043	35	23	#	100 0011	103	67	43	C	110 0011	143	99	63	c
010 0100	044	36	24	\$	100 0100	104	68	44	D	110 0100	144	100	64	d
010 0101	045	37	25	%	100 0101	105	69	45	E	110 0101	145	101	65	e
010 0110	046	38	26	&	100 0110	106	70	46	F	110 0110	146	102	66	f
010 0111	047	39	27	'	100 0111	107	71	47	G	110 0111	147	103	67	g
010 1000	050	40	28	(100 1000	110	72	48	H	110 1000	150	104	68	h
010 1001	051	41	29)	100 1001	111	73	49	I	110 1001	151	105	69	i
010 1010	052	42	2A	*	100 1010	112	74	4A	J	110 1010	152	106	6A	j
010 1011	053	43	2B	+	100 1011	113	75	4B	K	110 1011	153	107	6B	k
010 1100	054	44	2C	,	100 1100	114	76	4C	L	110 1100	154	108	6C	l
010 1101	055	45	2D	-	100 1101	115	77	4D	M	110 1101	155	109	6D	m
010 1110	056	46	2E	.	100 1110	116	78	4E	N	110 1110	156	110	6E	n
010 1111	057	47	2F	/	100 1111	117	79	4F	O	110 1111	157	111	6F	o
011 0000	060	48	30	0	101 0000	120	80	50	P	111 0000	160	112	70	p
011 0001	061	49	31	1	101 0001	121	81	51	Q	111 0001	161	113	71	q

Ref. Wikipedia

American Standard Code for Information Interchange (ASCII) 95 chart

DASL-100.2

C++ Programming and Linux

1. Data Type

```
Activities Text Editor Jan 29 16:09
datatype.cpp
1 #include <iostream>
2
3 int main() {
4
5     int age = 18;
6     float height = 5.7; // feet
7     double weight = 160.5;
8     char initial = 'C';
9     bool isMarried = false;
10
11     std::cout << "Age: " << age << std::endl;
12     std::cout << "Height: " << height << std::endl;
13     std::cout << "Weight: " << weight << std::endl;
14     std::cout << "Initial:" << initial << std::endl;
15     std::cout << "Married: " << isMarried << std::endl;
16
17     return 0;
18 };
19
```

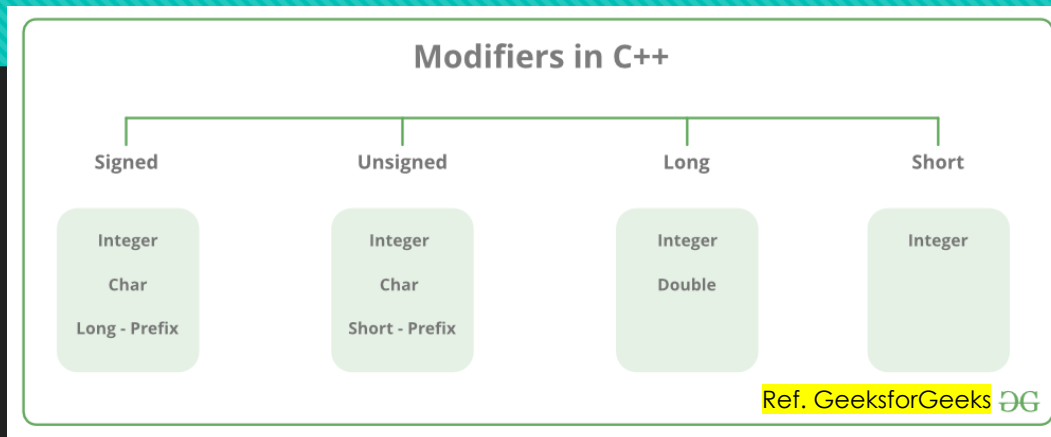
```
ubuntu20@ubuntu: ~
ubuntu20@ubuntu:~$ g++ datatype.cpp -o datatype
ubuntu20@ubuntu:~$ ls
datatype Desktop Downloads hello.cpp Pictures Templates
datatype.cpp Documents hello Music Public Videos
ubuntu20@ubuntu:~$ ./datatype
Age: 18
Height: 5.7
Weight: 160.5
Initial:C
Married: 0
ubuntu20@ubuntu:~$
```

DASL-100.2

C++ Programming and Linux

1. Data Type

- Modifier: specify the type and size of a data type, as well as its signedness, to ensure efficient and effective use of memory in a program.
 - short : modifies an integer type to occupy less memory (minimum 16 bits).
 - long: modifies an integer type to occupy more memory (minimum 32 bits).
 - signed: modifies an integer type to present positive and negative values (default for all integer types).
 - unsigned: modifies an integer type to represent only non-negative values (no negative values, positive and zero only).



Data Type	Size (in bytes)	Range
short int	2	-32,768 to 32,767
unsigned short int	2	0 to 65,535
unsigned int	4	0 to 4,294,967,295
int	4	-2,147,483,648 to 2,147,483,647
long int	4	-2,147,483,648 to 2,147,483,647
unsigned long int	8	0 to 4,294,967,295
long long int	8	-(2^63) to (2^63)-1
unsigned long long int	8	0 to 18,446,744,073,709,551,615
signed char	1	-128 to 127
unsigned char	1	0 to 255
float	4	
double	8	
long double	12	
wchar_t	2 or 4	1 wide character

Ref. GeeksforGeeks

DASL-100.2

C++ Programming and Linux

1. Data Type

- Modifier:

```

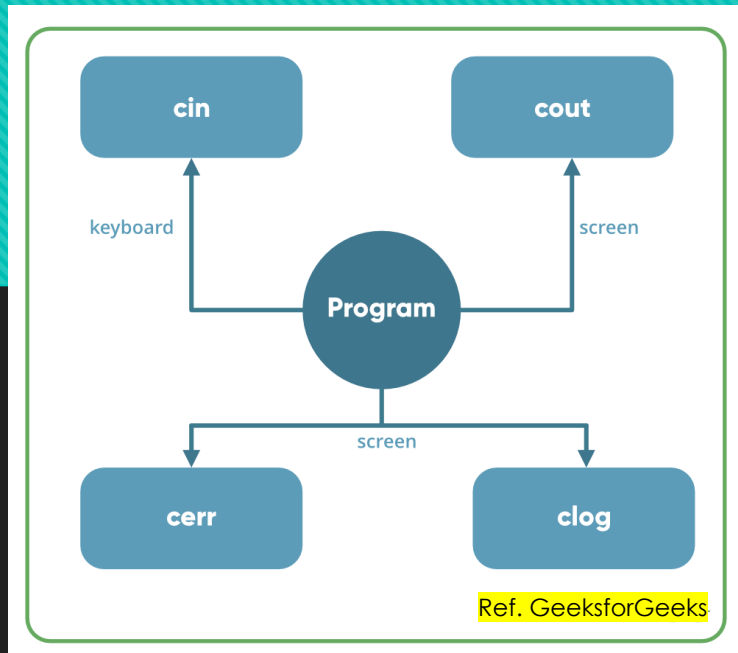
1 #include <ostream>
2
3 int main() {
4
5     //signed
6     signed int a = -10;
7     std::cout << "signed int a = " << a << std::endl;
8
9     //unsigned
10    unsigned int b = 10;
11    std::cout << "unsigned int b = " << b << std::endl;
12
13
14    //long
15    long int c = 123456789;
16    std::cout << "long int c = " << c << std::endl;
17
18    //short
19    short int d = 12345;
20    std::cout << "short int d = " << d << std::endl;
21
22    //long long
23    long long int e = 123456789012345;
24    std::cout << "long long int e = " << e << std::endl;
25
26    return 0;
27 };
    
```

```

ubuntu20@ubuntu:~$ g++ datatypemodifiers.cpp -o datatypemodifiers
ubuntu20@ubuntu:~$ ls
datatype          datatypemodifiers.cpp  Downloads  Music      Templates
datatype.cpp      Desktop                hello      Pictures   Videos
datatypemodifiers Documents              hello.cpp  Public
ubuntu20@ubuntu:~$ ./datatypemodifiers
signed int a = -10
unsigned int b = 10
long int c = 123456789
short int d = 12345
long long int e = 123456789012345
ubuntu20@ubuntu:~$
    
```

2. Data input and output:

- C++ input and output (I/O) refers to the communication of data between a program and input/output devices such as a keyboard, file, or display screen.
- The C++ standard library (iostream) provides several stream classes, including cin (standard input stream) and cout (standard output stream), that handle data input and output operations.



DASL-100.2

C++ Programming and Linux

2. Data input and output:

```
Activities | Text Editor | Feb 1 21:13 | datainputoutput.cpp | Save | - | □ | ×  
1 #include <iostream>  
2  
3 int main(){  
4  
5     int number;  
6     std::cout << "Enter an integer: ";  
7     std::cin >> number;  
8     std::cout << "You entered: " << number << std::endl;  
9  
10    return 0;  
11 };
```

C++ | Tab Width: 8 | Ln 10, Col 1 | INS

```
ubuntu20@ubuntu: ~ | Search | Menu | - | □ | ×  
ubuntu20@ubuntu:~$ g++ datainputoutput.cpp -o datainputoutput  
ubuntu20@ubuntu:~$ ls  
datainputoutput  datatypemodifiers  Downloads  Pictures  
datainputoutput.cpp  datatypemodifiers.cpp  hello  Public  
datatype  Desktop  hello.cpp  Templates  
datatype.cpp  Documents  Music  Videos  
ubuntu20@ubuntu:~$ ./datainputoutput  
Enter an integer: 5  
You entered: 5  
ubuntu20@ubuntu:~$
```


DASL-100.2

C++ Programming and Linux

3. Operators:

- In C++, operators are symbols that perform specific operations on one or more operands (values or variables) and produce a result.
- There are various types of operators in C++, including:
 - Arithmetic operators (e.g., +, -, *, /)
 - Comparison operators (e.g., ==, !=, >, <, >=, <=)
 - Logical operators (e.g., &&, ||, !)
 - Assignment operators (e.g., =, +=, -=, *=, /=)
 - Bitwise operators (e.g., &, |, ^, <<, >>)

DASL-100.2

C++ Programming and Linux

3. Operators:

Operator	Definition	Operator Type	Example
+	Adds two operands	Arithmetic	3 + 5 returns 8
-	Subtracts the second operand from the first	Arithmetic	7 - 4 returns 3
*	Multiplies two operands	Arithmetic	2 * 3 returns 6
/	Divides the first operand by the second	Arithmetic	9 / 3 returns 3
%	Returns the remainder of dividing the first operand by the second	Arithmetic	9 % 4 returns 1
==	Returns true if the two operands are equal, false otherwise	Comparison	3 == 5 returns false
!=	Returns true if the two operands are not equal, false otherwise	Comparison	3 != 5 returns true
>	Returns true if the first operand is greater than the second, false otherwise	Comparison	3 > 5 returns false

DASL-100.2

C++ Programming and Linux

3. Operators:

Operator	Definition	Operator Type	Example
<	Returns true if the first operand is less than the second, false otherwise	Comparison	3 < 5 returns true
>=	Returns true if the first operand is greater than or equal to the second, false otherwise	Comparison	3 >= 5 returns false
<=	Returns true if the first operand is less than or equal to the second, false otherwise	Comparison	3 <= 5 returns true
&&	Returns true if both operands are true, false otherwise	Logical	true && false returns false
	Returns true if either operand is true, false otherwise	Logical	`true
!	Returns the logical negation of the operand (i.e., true if the operand is false and vice versa)	Logical	!true returns false
=	Assigns the value of the right-side operand to the left-side operand	Assignment	x = 3 assigns the value 3 to x

DASL-100.2

C++ Programming and Linux

3. Operators:

Operator	Definition	Operator Type	Example
+=	Adds the right-side operand to the left-side operand and assigns the result to the left-side operand	Assignment	x += 3 adds 3 to x and assigns the result to x
-=	Subtracts the right-side operand from the left-side operand and assigns the result to the left-side operand	Assignment	x -= 3 subtracts 3 from x and assigns the result to x
*=	Multiplies the left-side operand by the right-side operand and assigns the result to the left-side operand	Assignment	x *= 3 multiplies x by 3 and assigns the result to x
/=	Divides the left-side operand by the right-side operand and assigns the result to the left-side operand	Assignment	x /= 3 divides x by 3 and assigns the result to x

DASL-100.2

C++ Programming and Linux

3. Operators:

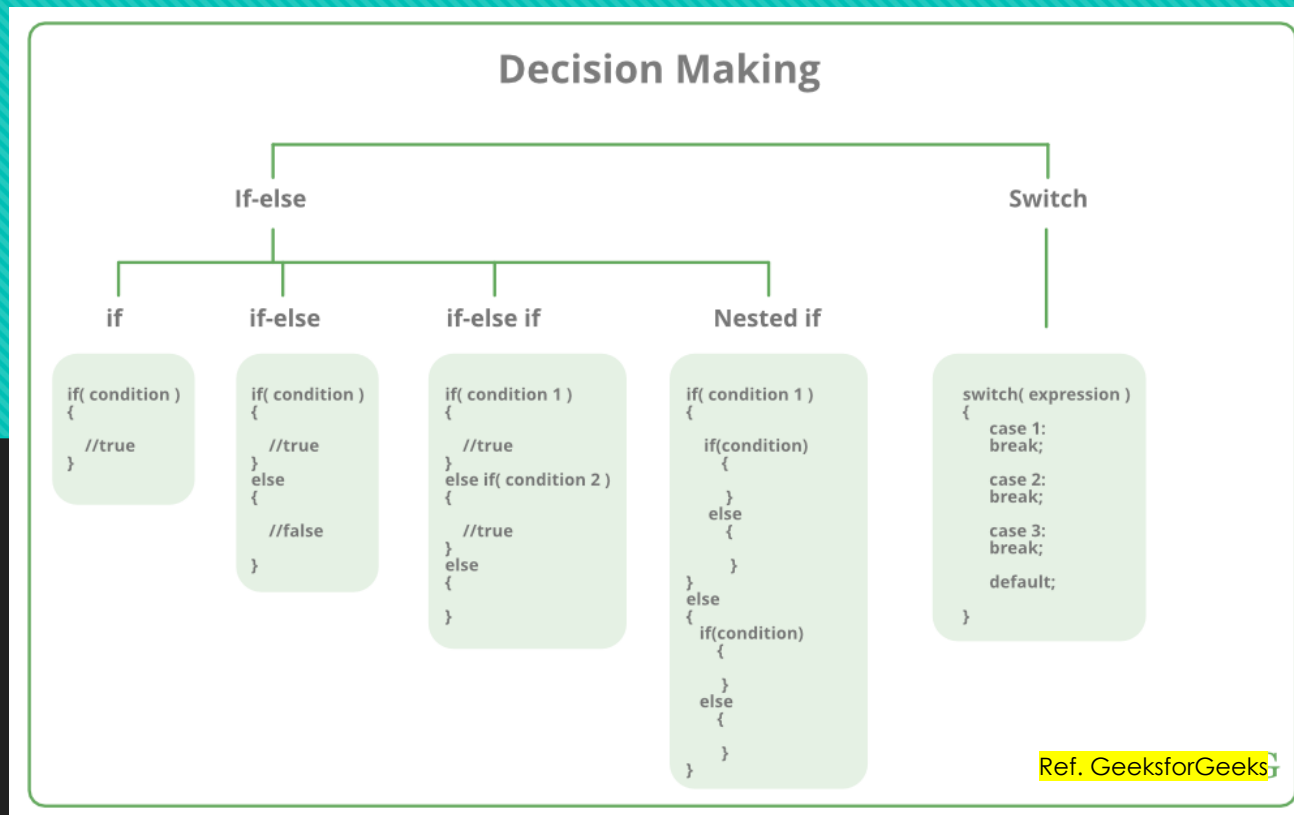
```
Activities | Text Editor | Feb 1 21:22 | operators.cpp | Save | [Menu] | [Close]
1 #include <iostream>
2
3 int main(){
4     int a = 5;
5     int b = 2;
6
7     std::cout << "a + b = " << a + b << std::endl;
8     std::cout << "a - b = " << a - b << std::endl;
9     std::cout << "a * b = " << a * b << std::endl;
10    std::cout << "a / b = " << a / b << std::endl;
11    std::cout << "a % b = " << a % b << std::endl;
12
13    return 0;
14 };
15
```

C++ | Tab Width: 8 | Ln 15, Col 1 | INS

```
ubuntu20@ubuntu: ~ | [Search] | [Menu] | [Close]
ubuntu20@ubuntu:~$ g++ operators.cpp -o operators
ubuntu20@ubuntu:~$ ls
datainputoutput  datatypemodifiers  Downloads  operators  Templates
datainputoutput.cpp  datatypemodifiers.cpp  hello  operators.cpp  Videos
datatype  Desktop  hello.cpp  Pictures
datatype.cpp  Documents  Music  Public
ubuntu20@ubuntu:~$ ./operators
a + b = 7
a - b = 3
a * b = 10
a / b = 2
a % b = 1
ubuntu20@ubuntu:~$
```

4. Decisions making:

- In C++, decision making refers to the process of making decisions based on certain conditions or expressions. The most commonly used decision making constructs are the if statement and the switch statement.



DASL-100.2

C++ Programming and Linux

4. Decisions making:

```

1 #include <iostream>
2
3 int main(){
4     int a = 5;
5     int b = 2;
6
7     if (a > b) {
8         std::cout << "a is greater than b" << std::endl;
9     } else {
10        std::cout << "b is greater than or equal to a " << std::endl;
11    }
12
13    return 0;
14 };
15

```

```

ubuntu20@ubuntu:~$ g++ ifelse.cpp -o ifelse
ubuntu20@ubuntu:~$ ls
datainputoutput      datatypemodifiers  Downloads  ifelse.cpp  Pictures
datainputoutput.cpp  datatypemodifiers.cpp  hello     Music      Public
datatype             Desktop            hello.cpp  operators   Templates
datatype.cpp         Documents          ifelse    operators.cpp  Videos
ubuntu20@ubuntu:~$ ./ifelse
a is greater than b
ubuntu20@ubuntu:~$

```

DASL-100.2

C++ Programming and Linux

4. Decisions making:

```

1 #include <iostream>
2
3 int main(){
4
5     int day = 4;
6
7     switch(day) {
8         case 1:
9             std::cout << "Monday" << std::endl;
10            break;
11        case 2:
12            std::cout << "Tuesday" << std::endl;
13            break;
14        case 3:
15            std::cout << "Wednesday" << std::endl;
16            break;
17        case 4:
18            std::cout << "Thursday" << std::endl;
19            break;
20        case 5:
21            std::cout << "Friday" << std::endl;
22            break;
23        case 6:
24            std::cout << "Saturday" << std::endl;
25            break;
26        case 7:
27            std::cout << "Sunday" << std::endl;
28            break;
29        default:
30            std::cout << "Invalid day" << std::endl;
31    };
32
33    return 0;
34 };
    
```

```

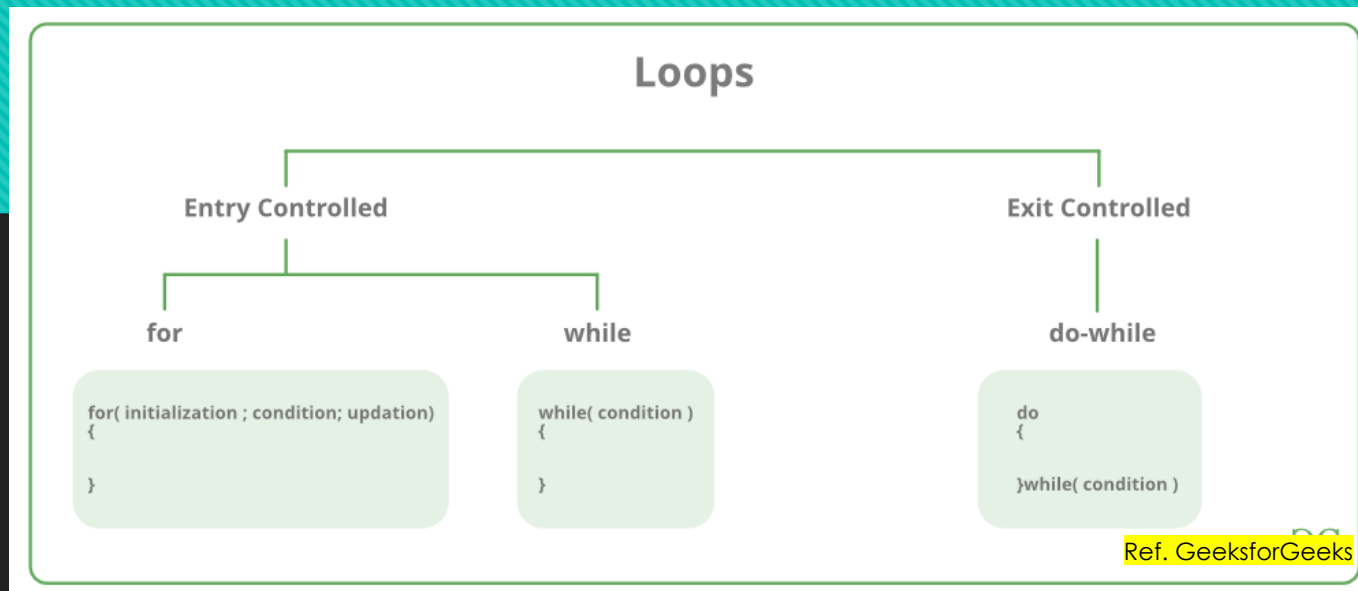
ubuntu20@ubuntu:~$ g++ switchcase.cpp -o switchcase
ubuntu20@ubuntu:~$ ls
datainputoutput      Desktop      ifelse.cpp      switchcase
datainputoutput.cpp  Documents   Music           switchcase.cpp
datatype             Downloads   operators       Templates
datatype.cpp         hello      operators.cpp   Videos
datatypemodifiers    hello.cpp   Pictures
datatypemodifiers.cpp ifelse     Public
ubuntu20@ubuntu:~$ ./switchcase
Thursday
ubuntu20@ubuntu:~$
    
```


DASL-100.2

C++ Programming and Linux

5. Loops:

- In C++, there are two types of loops: entry-controlled loops and exit-controlled loops.
- An entry-controlled loop is a loop where the condition to continue the loop is tested before each iteration. The **for** and **while** loops are examples of entry-controlled loops.
- An exit-controlled loop is a loop where the condition to continue the loop is tested after each iteration. The **do-while loop** is an example of an exit-controlled loop.



DASL-100.2

C++ Programming and Linux

5. Loops:

- "for" loop:

```
Activities | Text Editor | Feb 1 22:00 | forloop.cpp | Save | - | + | x
1 #include <iostream>
2
3 int main(){
4     for (int i = 0; i < 5; i++) {
5         std::cout << "Interation: " << i << std::endl;
6     };
7
8     return 0;
9 };
```

C++ | Tab Width: 8 | Ln 9, Col 3 | INS

```
ubuntu20@ubuntu: ~ | Search | Menu | - | + | x
ubuntu20@ubuntu:~$ g++ forloop.cpp -o forloop
ubuntu20@ubuntu:~$ ls
datainputoutput      Desktop      hello.cpp      Pictures
datainputoutput.cpp  Documents   ifelse         Public
datatype             Downloads   ifelse.cpp     switchcase
datatype.cpp          Music       ifelse.cpp     switchcase.cpp
datatypemodifiers    forloop.cpp operators       Templates
datatypemodifiers.cpp hello        operators.cpp  Videos
ubuntu20@ubuntu:~$ ./forloop
Interation: 0
Interation: 1
Interation: 2
Interation: 3
Interation: 4
ubuntu20@ubuntu:~$
```

DASL-100.2

C++ Programming and Linux

5. Loops:

- "while" loop:

```
Activities Text Editor Feb 1 22:02 whileloop.cpp Save
1 #include <iostream>
2
3 int main(){
4
5     int i = 0;
6     while (i < 5) {
7         std::cout << "Iteration : " << i << std::endl;
8         i++;
9     };
10
11     return 0;
12
13 };
```

```
ubuntu20@ubuntu: ~
ubuntu20@ubuntu:~$ g++ whileloop.cpp -o whileloop
ubuntu20@ubuntu:~$ ls
datainputoutput      Documents      ifelse.cpp      switchcase.cpp
datainputoutput.cpp  Downloads     Music           Templates
datatype             forloop       operators       Videos
datatype.cpp         forloop.cpp   operators.cpp   whileloop
datatypemodifiers   hello        Pictures        whileloop.cpp
datatypemodifiers.cpp hello.cpp     Public
Desktop             ifelse       switchcase
ubuntu20@ubuntu:~$ ./whileloop
Iteration : 0
Iteration : 1
Iteration : 2
Iteration : 3
Iteration : 4
ubuntu20@ubuntu:~$
```

DASL-100.2

C++ Programming and Linux

5. Loops:

- "do-while" loop:

```

1 #include <iostream>
2
3 int main(){
4     int i = 0;
5     do {
6         std::cout << "Iteration :" << i << std::endl;
7         i++;
8     } while (i < 5);
9
10    return 0;
11
12 };|
    
```

```

ubuntu20@ubuntu:~$ g++ dowhileloop.cpp -o dowhileloop
ubuntu20@ubuntu:~$ ls
datainputoutput      Documents      hello.cpp       Public
datainputoutput.cpp  dowhileloop   ifelse          switchcase
datatype              dowhileloop.cpp  ifelse.cpp     switchcase.cpp
datatype.cpp          Downloads      Music           Templates
datatypemodifiers    forloop       operators       Videos
datatypemodifiers.cpp forloop.cpp    operators.cpp   whileloop
Desktop              hello         Pictures        whileloop.cpp
ubuntu20@ubuntu:~$ ./dowhileloop
Iteration :0
Iteration :1
Iteration :2
Iteration :3
Iteration :4
ubuntu20@ubuntu:~$
    
```

6. Debugging:

- Debugging in C++ is the process of finding and fixing errors in a program's source code to make it run correctly. This can be done manually or using a debugger tool that allows you to step through the code, inspect variables, and set breakpoints. Debugging is an important part of the software development process to ensure that the program behaves as expected.

```
ubuntu20@ubuntu: ~  
__s)  
|  
/usr/include/c++/9/ostream:583:5: note: template argument deduction/substitution failed:  
whileloop.cpp:7:44: note: cannot convert 'std::end' (type '<unresolved overloaded function type>') to type 'const unsigned char*'   
7 | std::cout << "Iteration : " << i << std::end;  
|  
In file included from /usr/include/c++/9/iostream:39,  
from whileloop.cpp:1:  
/usr/include/c++/9/ostream:691:5: note: candidate: 'template<class _Ostream, class _Tp> typename std::enable_if<std::__and<std::__not<std::is_lvalue_reference<_Tp>>, std::__is_convertible_to_basic_ostream<_Ostream>, std::__is_insertable<typename std::__is_convertible_to_basic_ostream<_Tp>::__ostream_type, const _Tp&, void>>::value, typename std::__is_convertible_to_basic_ostream<_Tp>::__ostream_type>::type std::operator<<(_Ostream&&, const _Tp&)'   
691 | operator<<(_Ostream&& __os, const _Tp& __x)  
|  
/usr/include/c++/9/ostream:691:5: note: template argument deduction/substitution failed:  
whileloop.cpp:7:44: note: couldn't deduce template parameter '_Tp'   
7 | std::cout << "Iteration : " << i << std::end;  
|  
ubuntu20@ubuntu:~$
```

DASL-100.2

C++ Programming and Linux

6. Debugging:

- “Nano” is a text editor for the command line in Linux. It is a simple, user-friendly editor that can be used to create and edit text files.
- Example : nano forloop.cpp.
- The bottom of the screen displays information about the available commands. For example, to save changes to the file, you can press **Ctrl + O** and then press Enter. To exit nano, you can press **Ctrl + X**.

```

ubuntu20@ubuntu: ~
__s)
|
|_ /usr/include/c++/9/ostream:583:5: note: template argument deduction/substitution
on failed:
forloop.cpp:5:44: note: cannot convert 'std::end' (type '<unresolved overloaded
function type>') to type 'const unsigned char*'
5 | std::cout << "Iteration: " << i << std::end;
|
|_ In file included from /usr/include/c++/9/iostream:39,
from forloop.cpp:1:
/usr/include/c++/9/ostream:691:5: note: candidate: 'template<class _Ostream, cla
ss _Tp> typename std::enable_if<std::__and<std::__not<std::is_lvalue_reference
<_Tp> >, std::__is_convertible_to_basic_ostream<_Ostream>, std::__is_insertable<
typename std::__is_convertible_to_basic_ostream<_Tp>::__ostream_type, const _Tp&
, void> >::value, typename std::__is_convertible_to_basic_ostream<_Tp>::__ostrea
m_type>::type std::operator<<(_Ostream&&, const _Tp&)
691 | operator<<(_Ostream&& __os, const _Tp& __x)
|
|_ /usr/include/c++/9/ostream:691:5: note: template argument deduction/substitutio
on failed:
forloop.cpp:5:44: note: couldn't deduce template parameter '_Tp'
5 | std::cout << "Iteration: " << i << std::end;
|
|_
ubuntu20@ubuntu:~$ nano forloop.cpp

```

```

GNU nano 4.8 forloop.cpp
#include <iostream>

int main(){
    for (int i = 0; i < 5; i++) {
        std::cout << "Iteration: " << i << std::end;
    };

    return 0;
};

Read 9 lines
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line

```