UNLV Howard R. Hughes College of Engineering

DASL-100.2
C++ Programming and Linux

DASL
Drones and Autonomous Systems Lab @ UNLV

Week 2-1

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

Instructor: Truc Tran

02-04-2023

UNLV Howard R. Hughes College of Engineering

DASL-100.2 C++ Programming and Linux

DASL Drones and Autonomous Systems Lab @ UNLV

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)

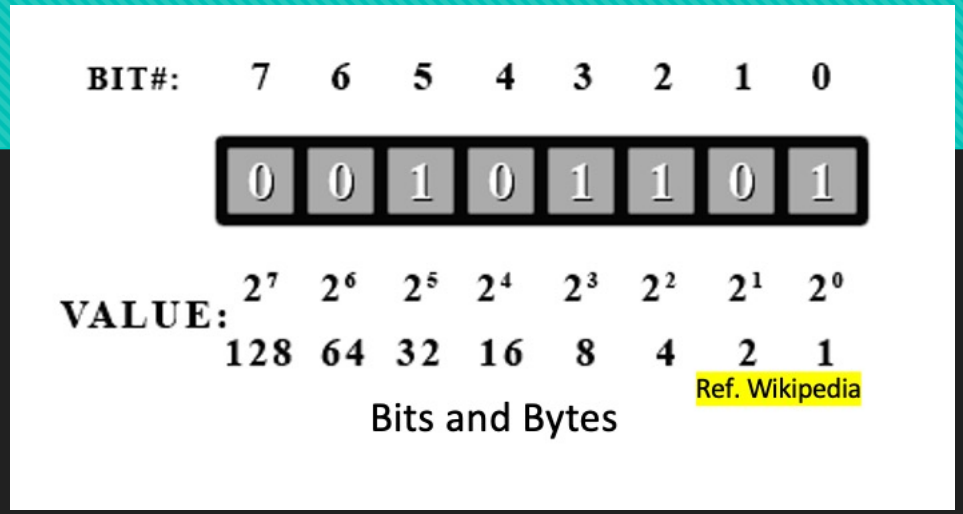
UNLV Howard R. Hughes College of ENGINEERING

DASL-100.2 C++ Programming and Linux

DASL
Drones and Autonomous Systems Lab @ UNLV

1. Data Type

- *Memory space occupied by each data type depends on the complier 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:



Bits and Bytes

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	16			

Ref. Wikipedia

American Standard Code for Information Interchange (ASCII) 95 chart

UNLV Howard R. Hughes College of Engineering

DASL-100.2 C++ Programming and Linux

DASL Drones and Autonomous Systems Lab @ UNLV

1. Data Type

Activities Text Editor Jan 29 16:09

Open datatype.cpp Save

```cpp
#include <iostream>

int main() {

        int age = 18;
        float height = 5.7; // feet
        double weight = 160.5;
        char initial  = 'C';
        bool isMarried = false;

        std::cout << "Age: " << age << std::endl;
        std::cout << "Height: " << height << std::endl;
        std::cout << "Weight: " << weight << std::endl;
        std::cout << "Initial:" << initial  << std::endl;
        std::cout << "Married: " << isMarried << std::endl;

        return 0;
};
```

C++ Tab Width: 8 Ln 19, Col 9 INS

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:~$
```

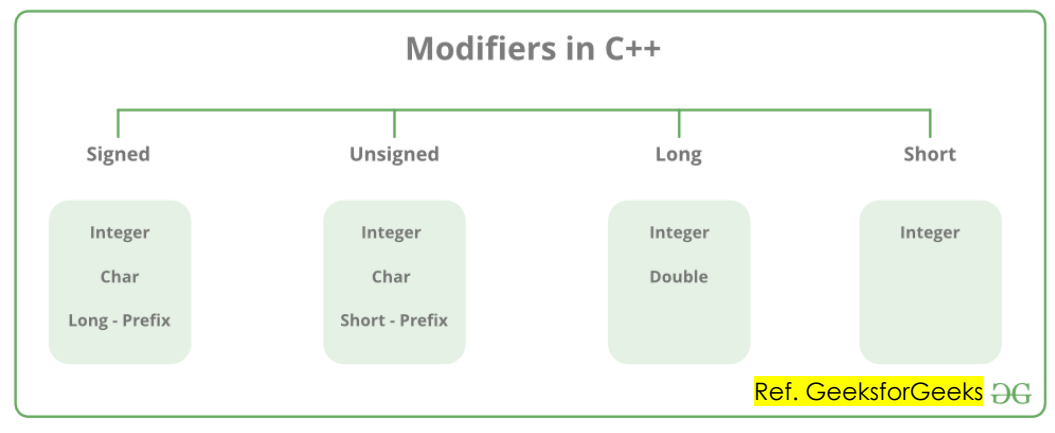
UNLV Howard R. Hughes College of Engineering

DASL-100.2 C++ Programming and Linux

DASL Drones and Autonomous Systems Lab @ UNLV

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

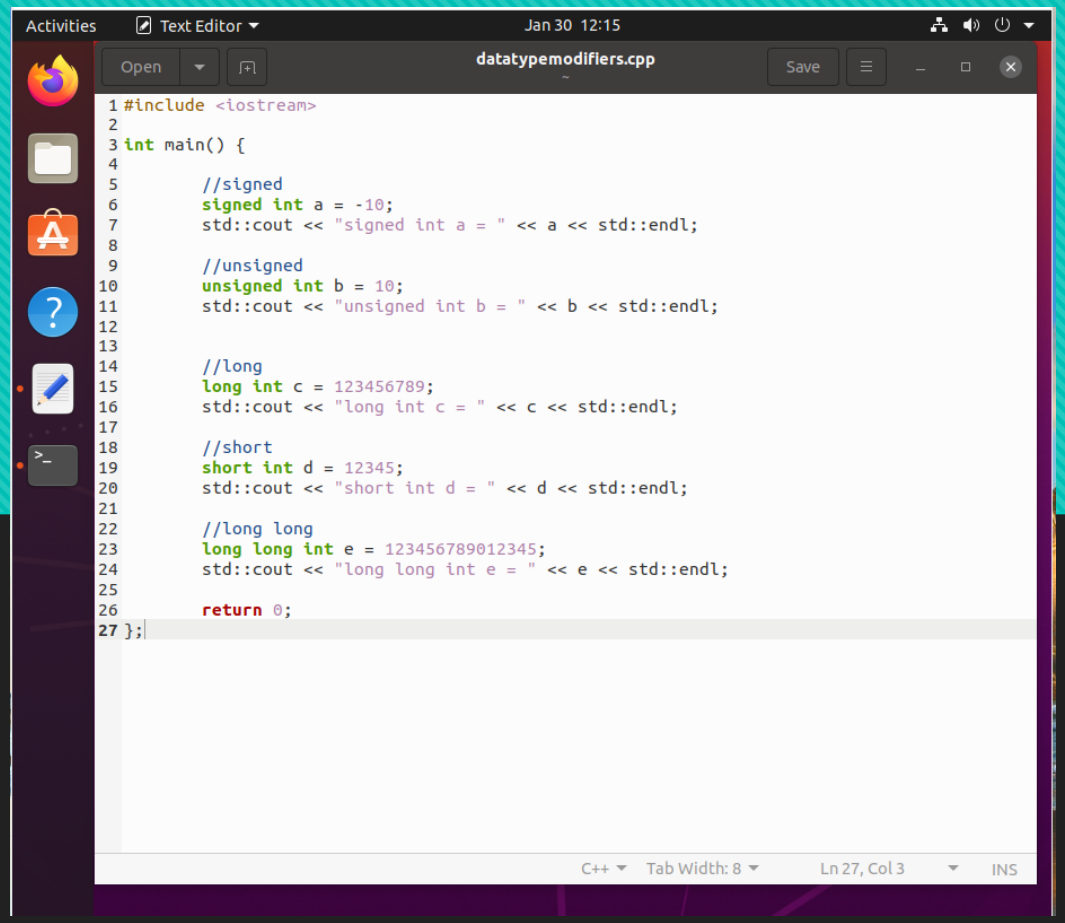
UNLV Howard R. Hughes College of Engineering

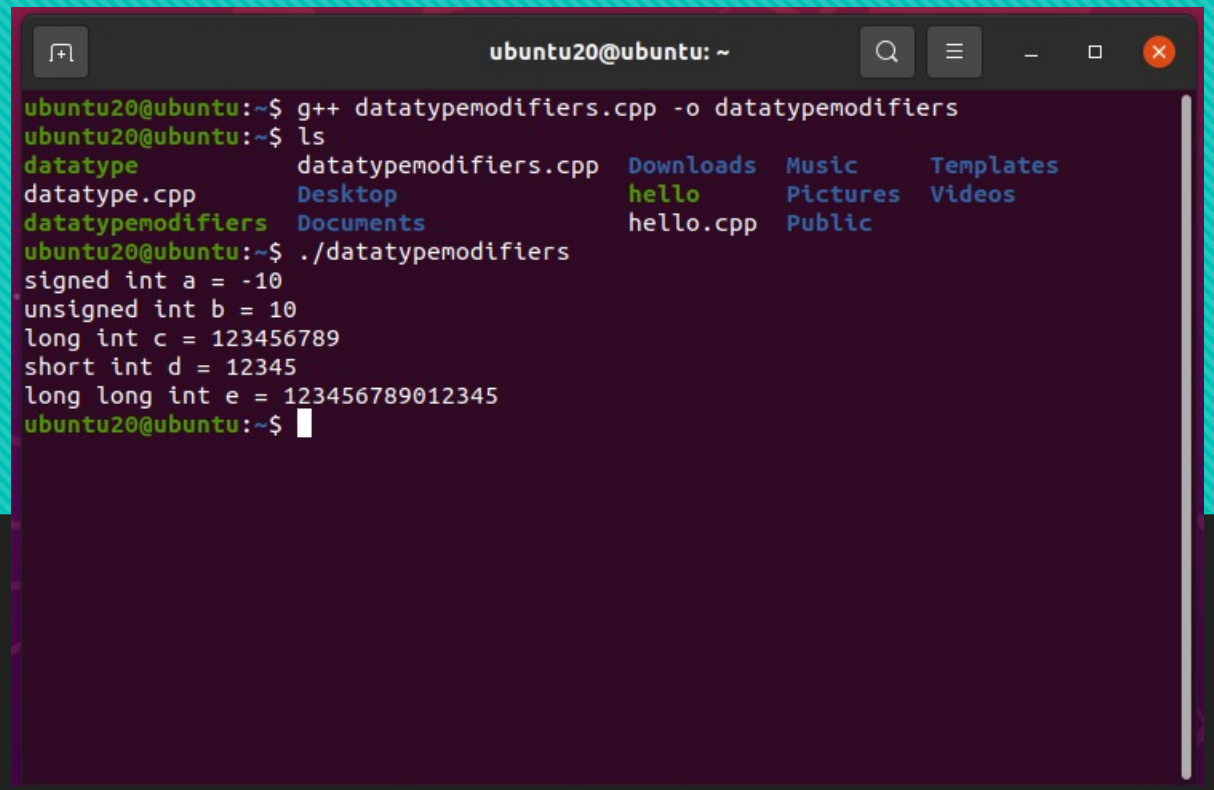
DASL-100.2 C++ Programming and Linux

DASL Drones and Autonomous Systems Lab @ UNLV

1. Data Type

- Modifier:





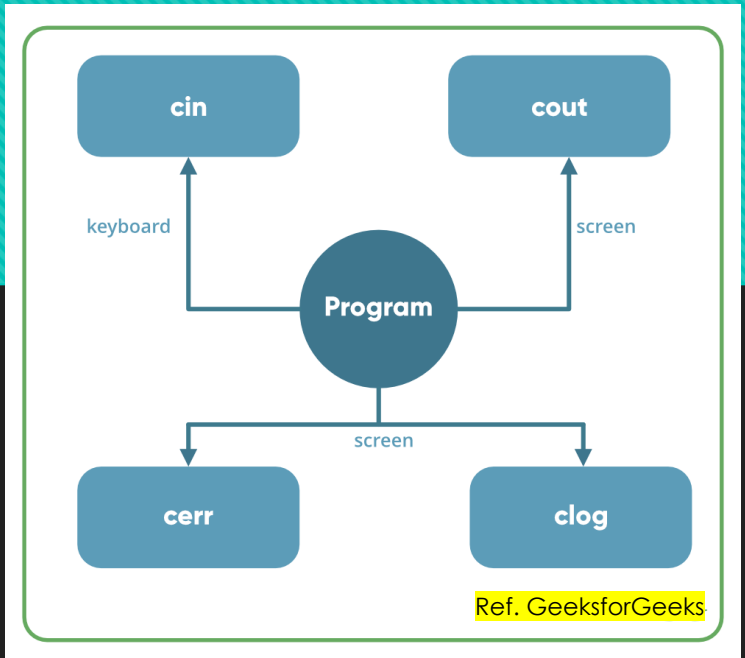
UNLV Howard R. Hughes College of ENGINEERING

DASL-100.2 C++ Programming and Linux

DASL Drones and Autonomous Systems Lab @ UNLV

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.

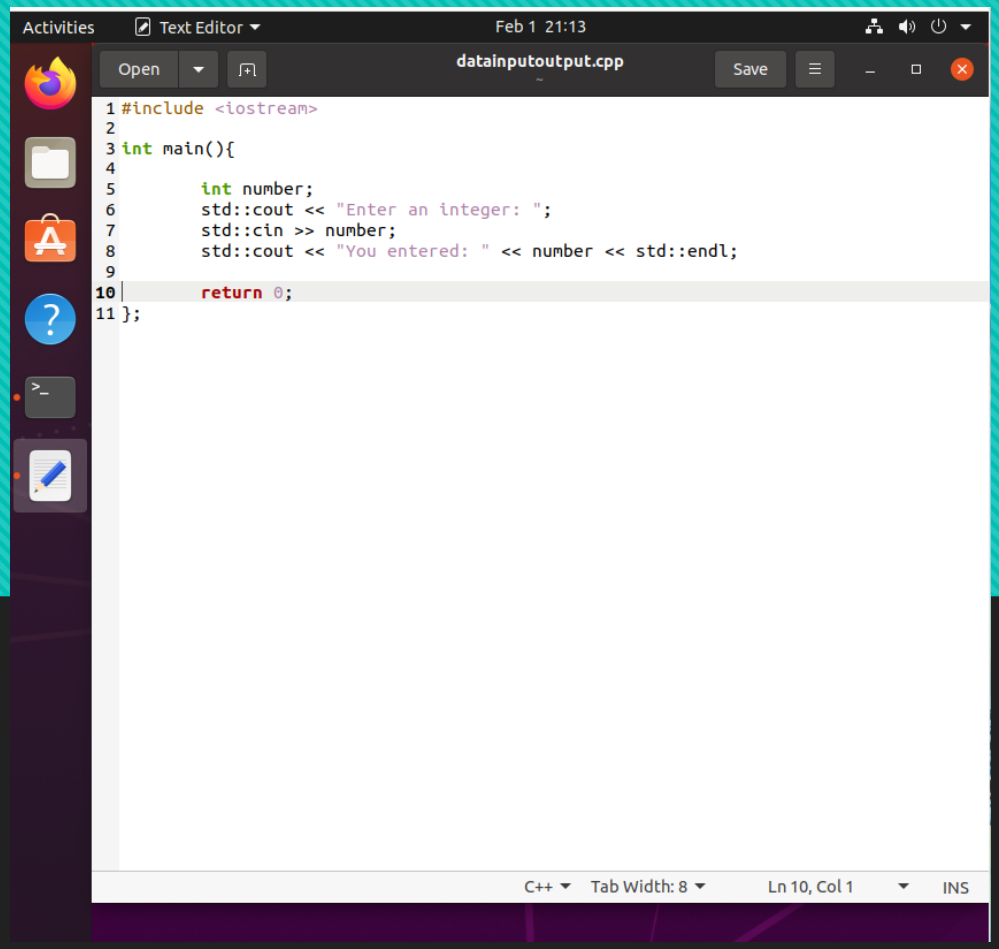


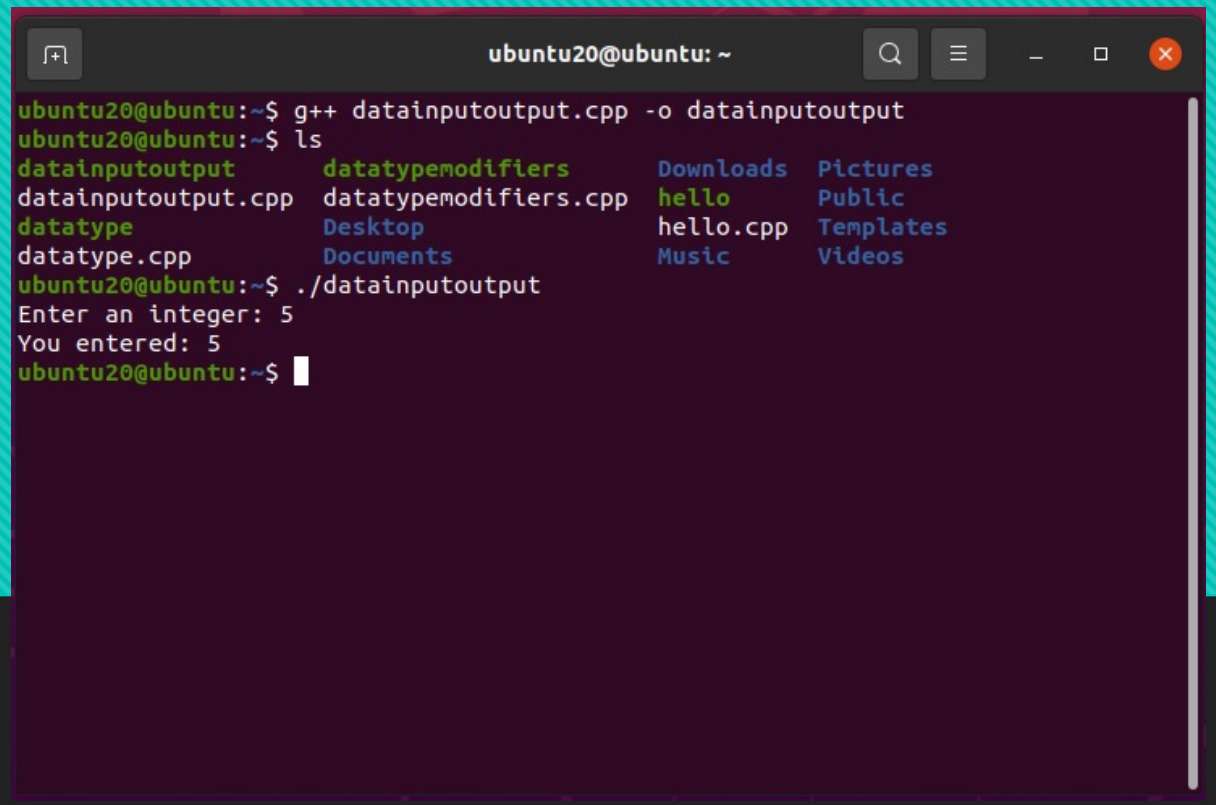
UNLV Howard R. Hughes College of Engineering

DASL-100.2 C++ Programming and Linux

DASL Drones and Autonomous Systems Lab @ UNLV

2. Data input and output:





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:

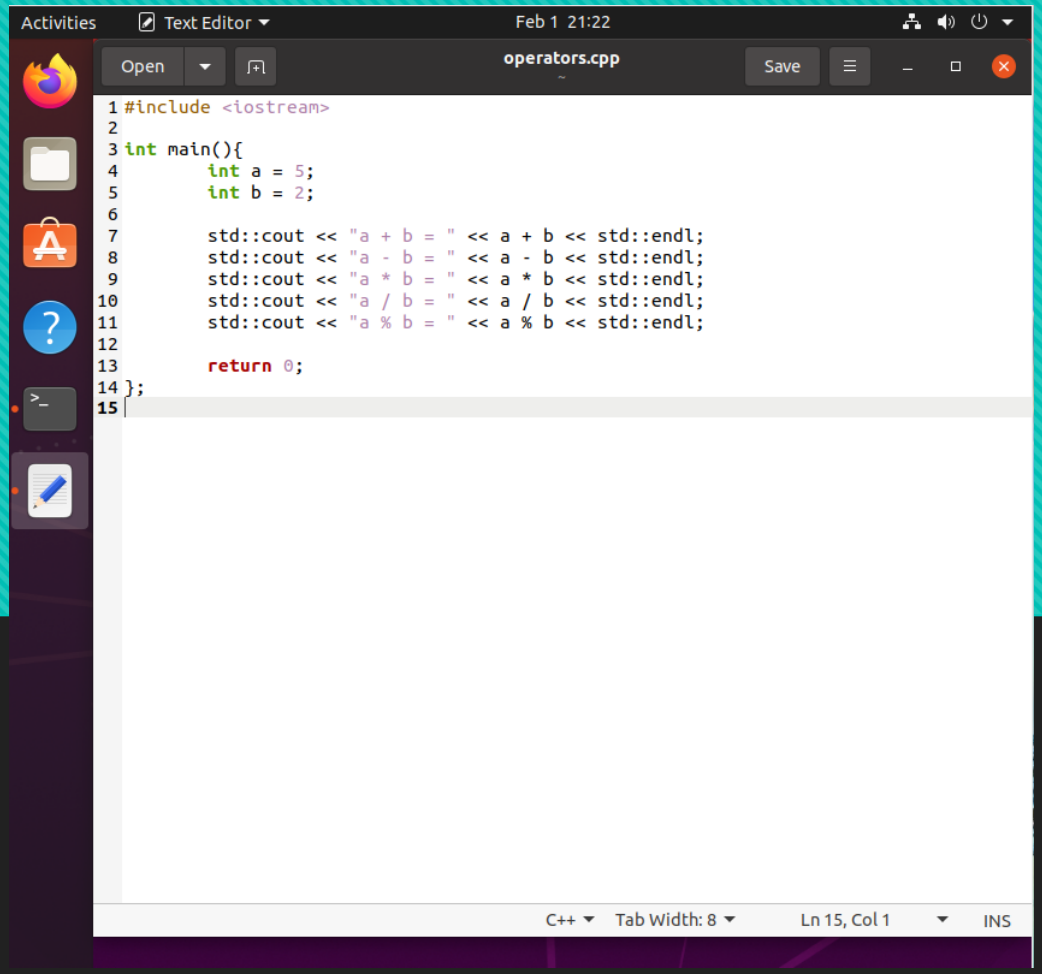
Operator	Definition	Operator Type	Example
+=	Adds the right-side operand to the left-side operand and assigns the result to the left-side operand	Assignment	x += 3 adds 3 to x and assigns the result to x
-=	Subtracts the right-side operand from the left-side operand and assigns the result to the left-side 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

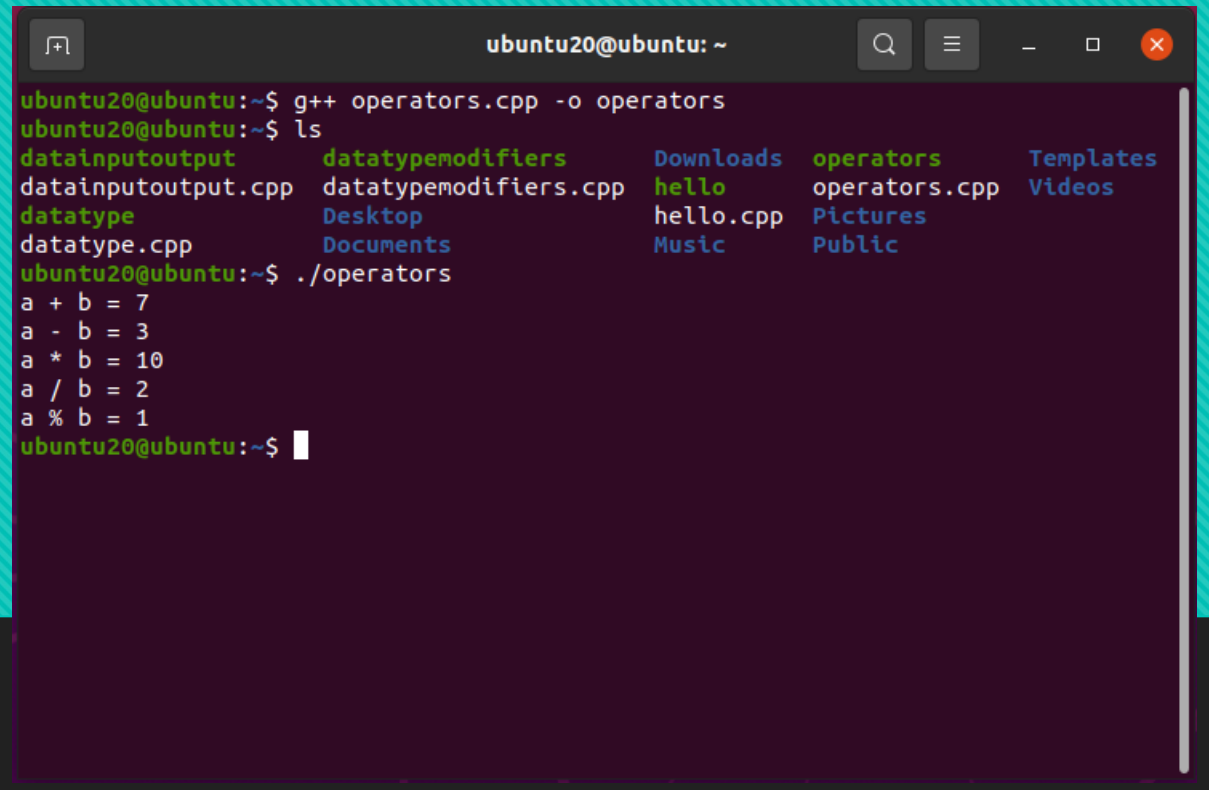
UNLV Howard R. Hughes College of Engineering

DASL-100.2 C++ Programming and Linux

DASL Drones and Autonomous Systems Lab @ UNLV

3. Operators:





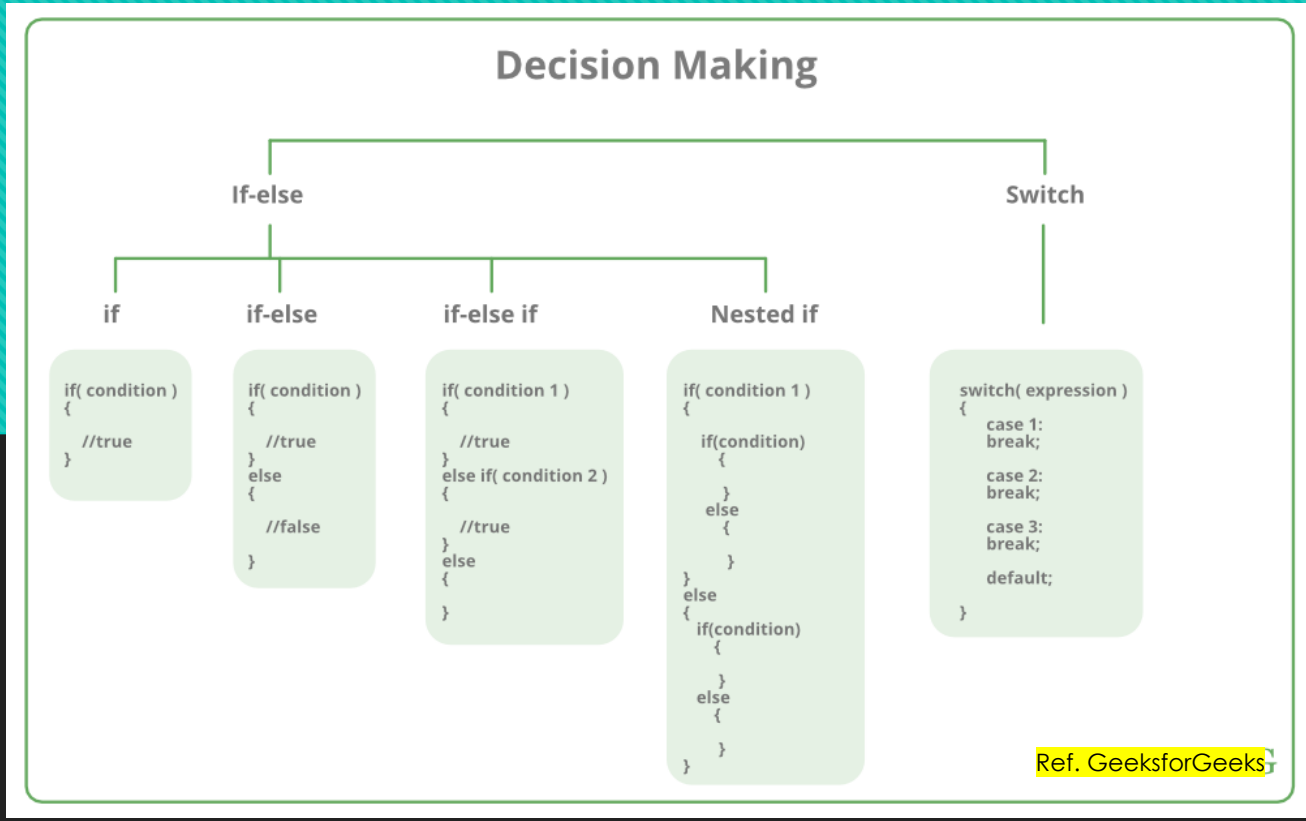
UNLV Howard R. Hughes College of ENGINEERING

DASL-100.2 C++ Programming and Linux

DASL Drones and Autonomous Systems Lab @ UNLV

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**.

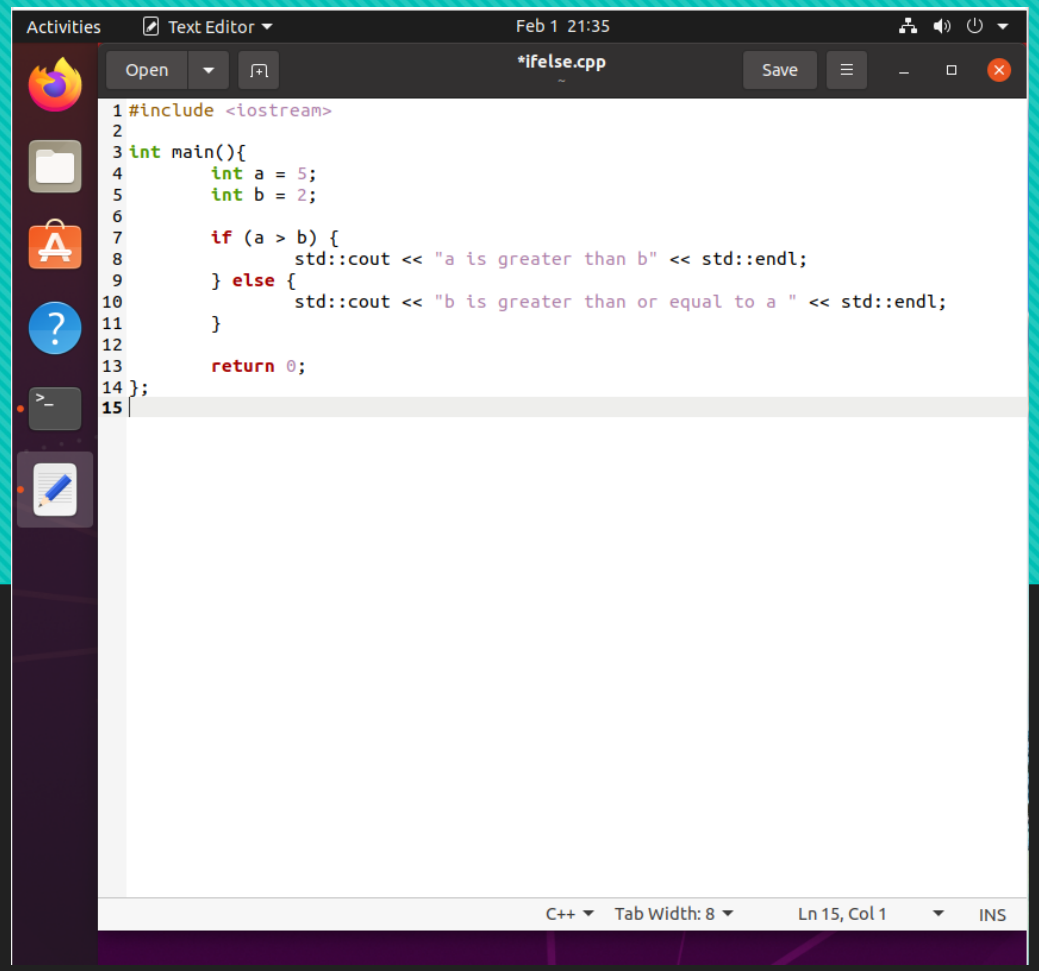


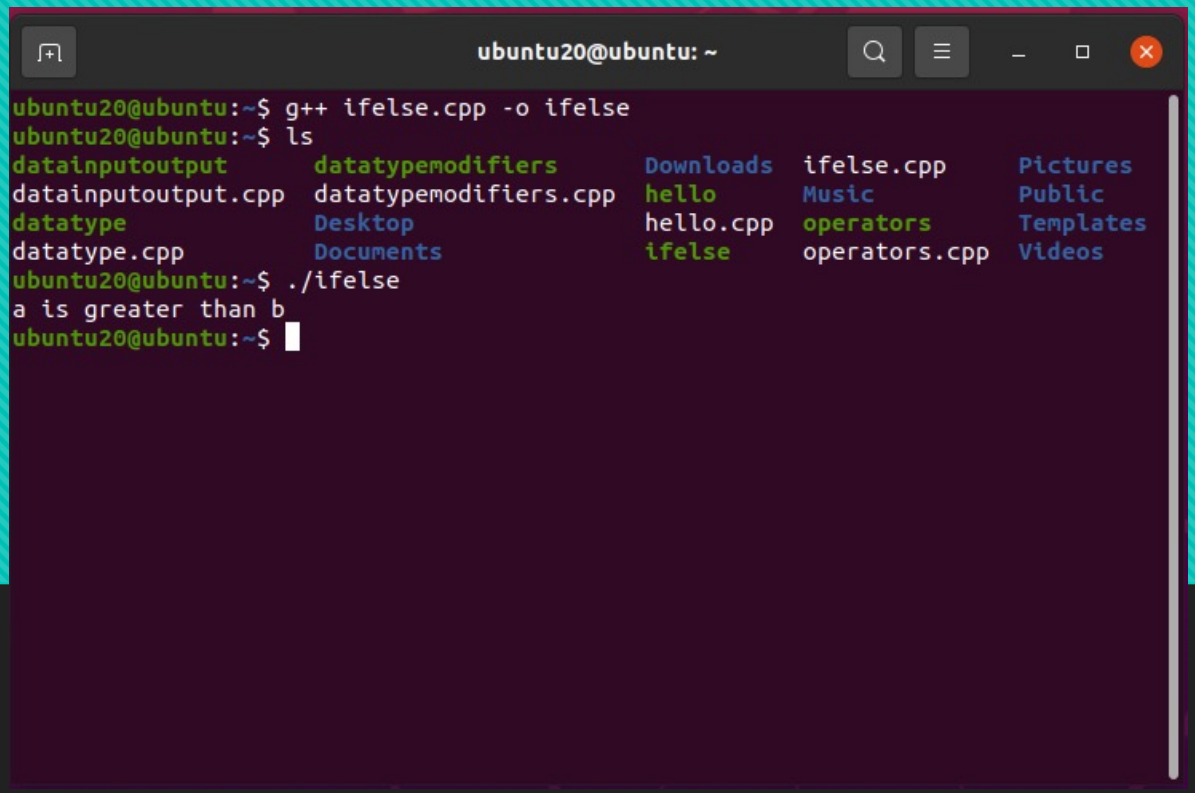
UNLV Howard R. Hughes College of Engineering

DASL-100.2 C++ Programming and Linux

DASL Drones and Autonomous Systems Lab @ UNLV

4. Decisions making:



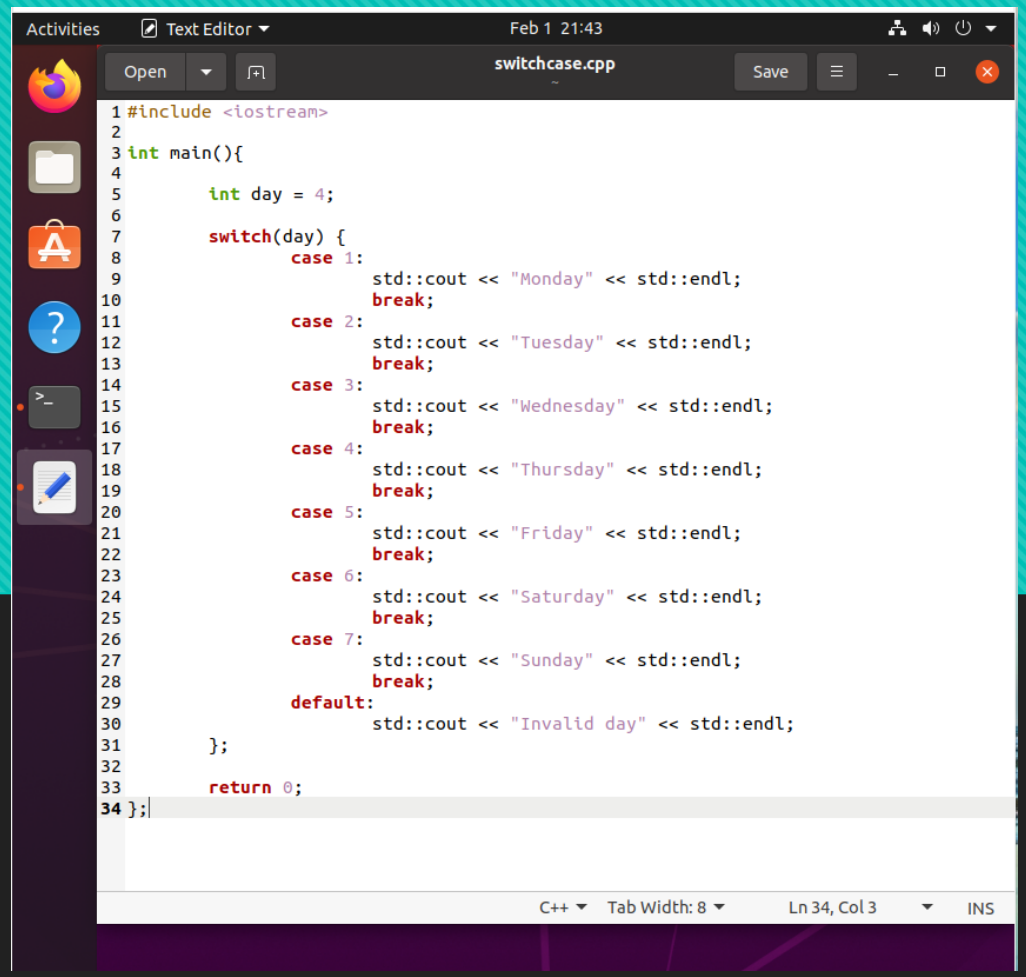


UNLV Howard R. Hughes College of Engineering

DASL-100.2 C++ Programming and Linux

DASL Drones and Autonomous Systems Lab @ UNLV

4. Decisions making:



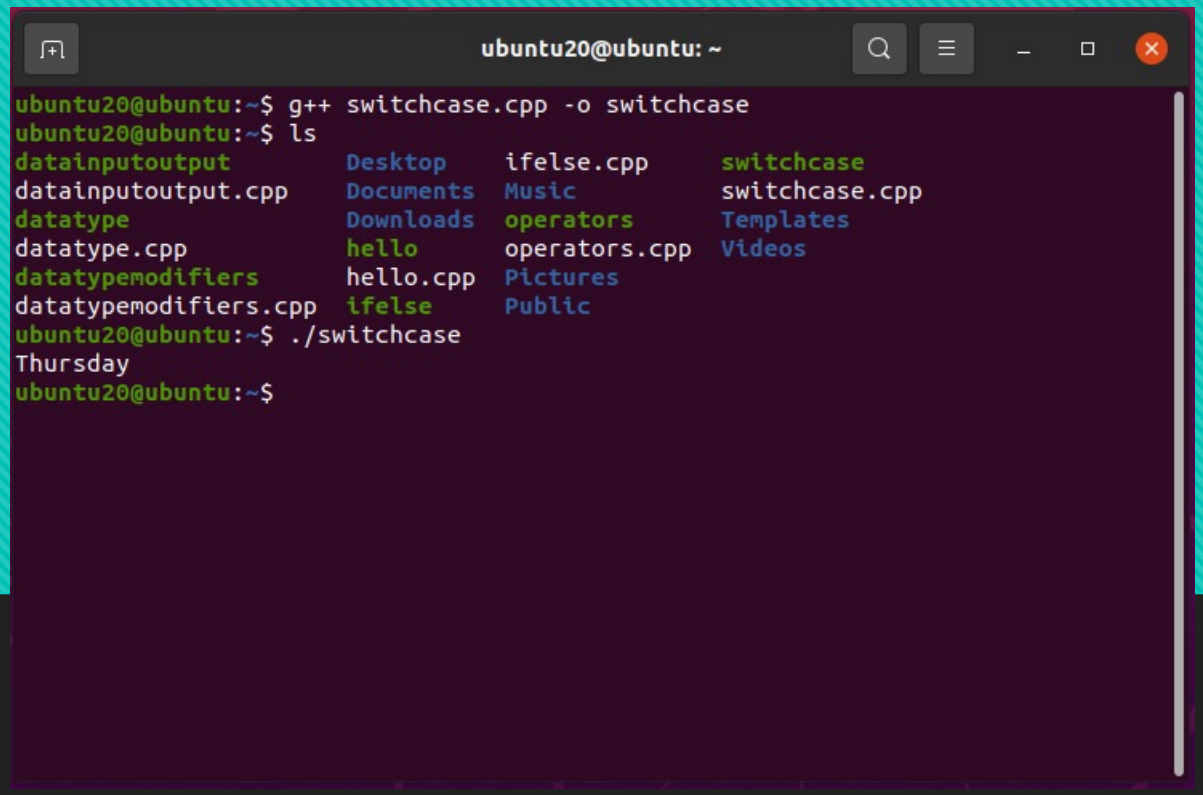

```cpp
#include <iostream>

int main(){

        int day = 4;

        switch(day) {
                case 1:
                        std::cout << "Monday" << std::endl;
                        break;
                case 2:
                        std::cout << "Tuesday" << std::endl;
                        break;
                case 3:
                        std::cout << "Wednesday" << std::endl;
                        break;
                case 4:
                        std::cout << "Thursday" << std::endl;
                        break;
                case 5:
                        std::cout << "Friday" << std::endl;
                        break;
                case 6:
                        std::cout << "Saturday" << std::endl;
                        break;
                case 7:
                        std::cout << "Sunday" << std::endl;
                        break;
                default:
                        std::cout << "Invalid day" << std::endl;
        };

        return 0;
};
```



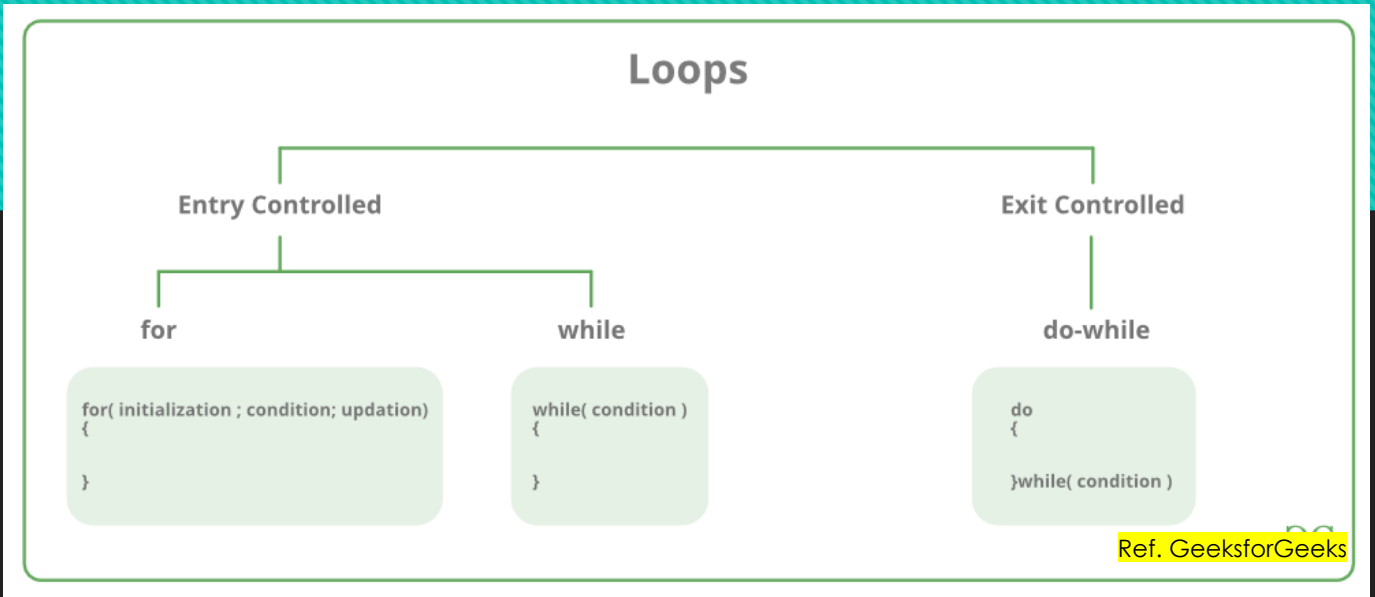
UNLV Howard R. Hughes College of Engineering

DASL-100.2 C++ Programming and Linux

DASL Drones and Autonomous Systems Lab @ UNLV

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.



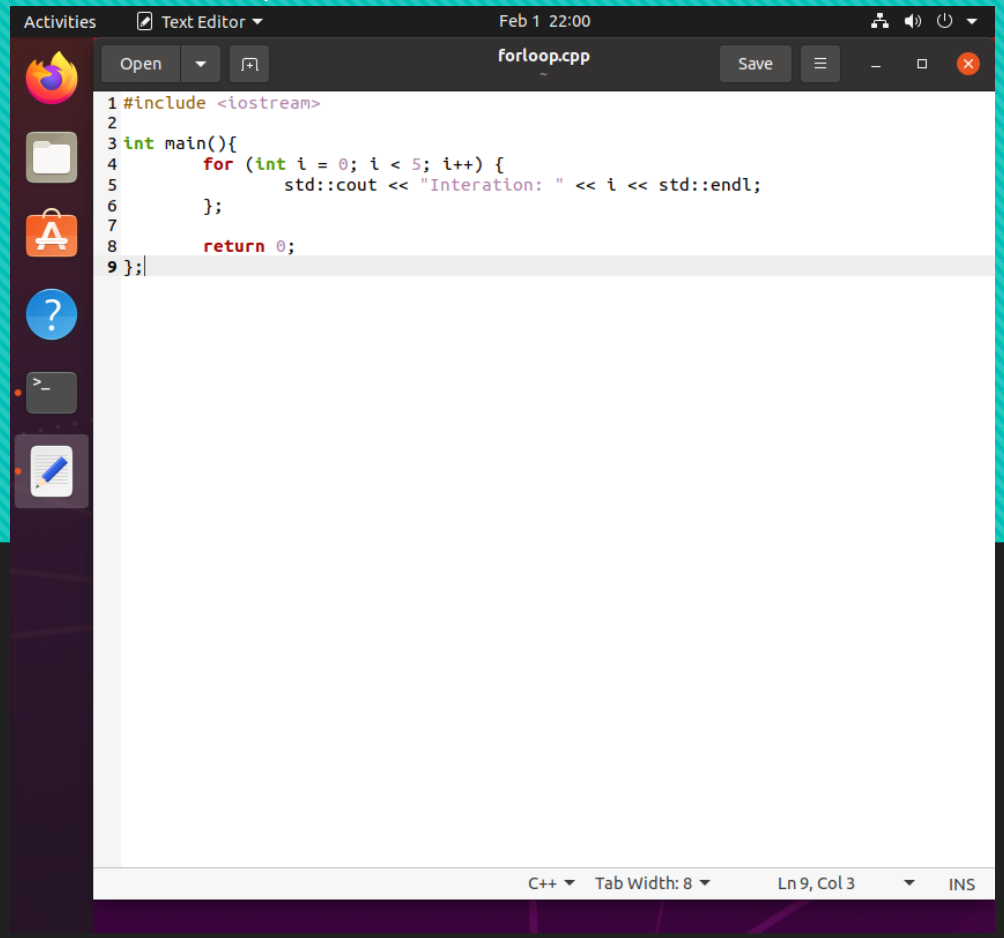
UNLV Howard R. Hughes College of Engineering

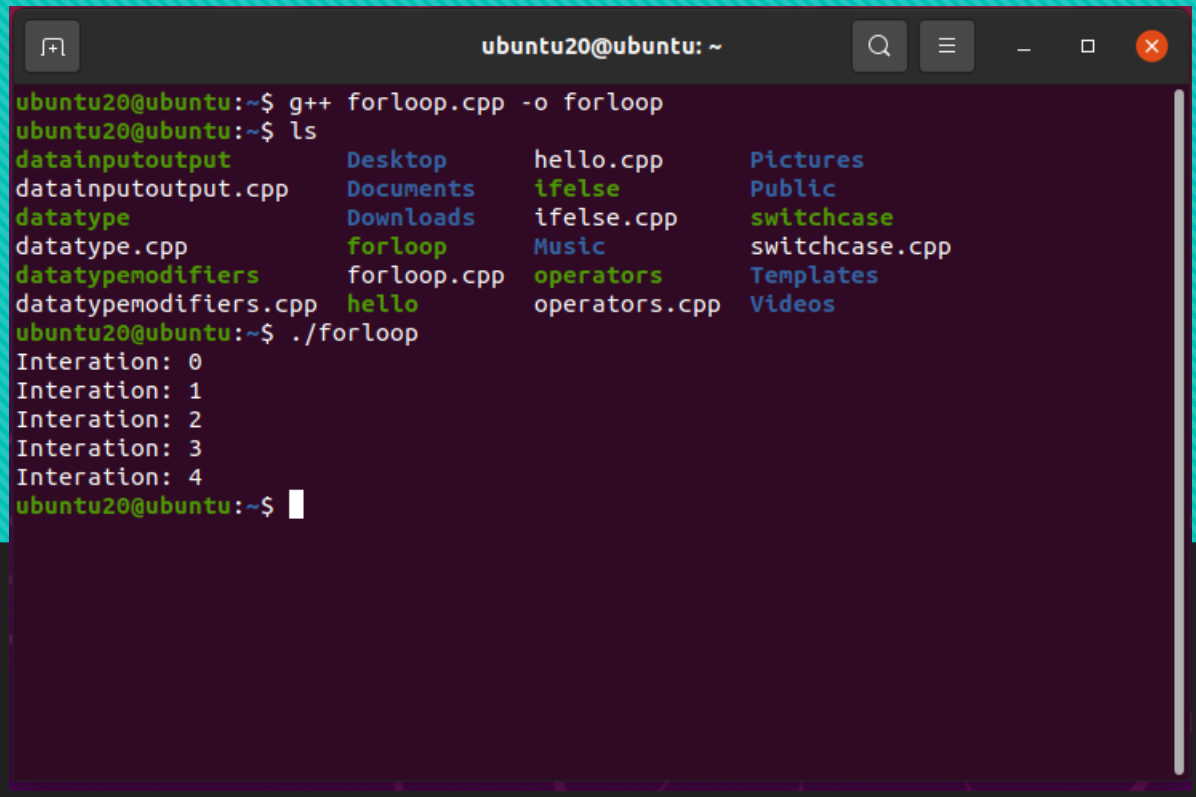
DASL-100.2 C++ Programming and Linux

DASL Drones and Autonomous Systems Lab @ UNLV

5. Loops:

- "for" loop:





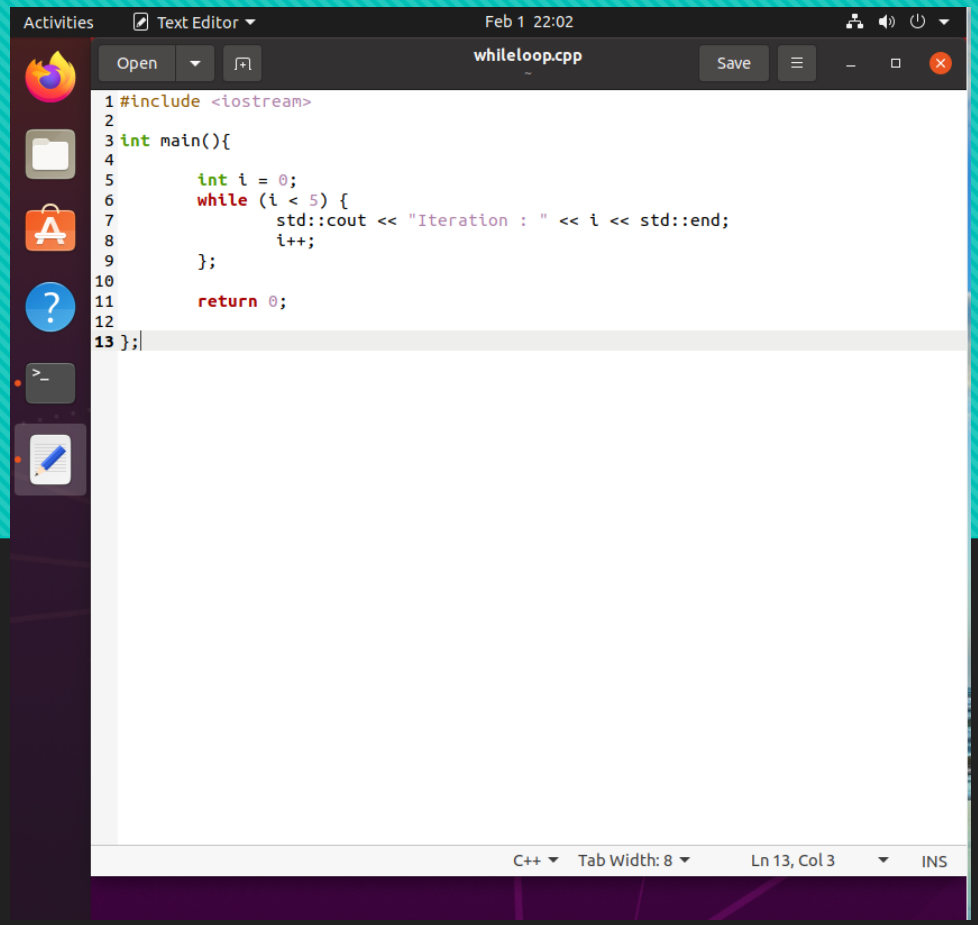
UNLV Howard R. Hughes College of Engineering

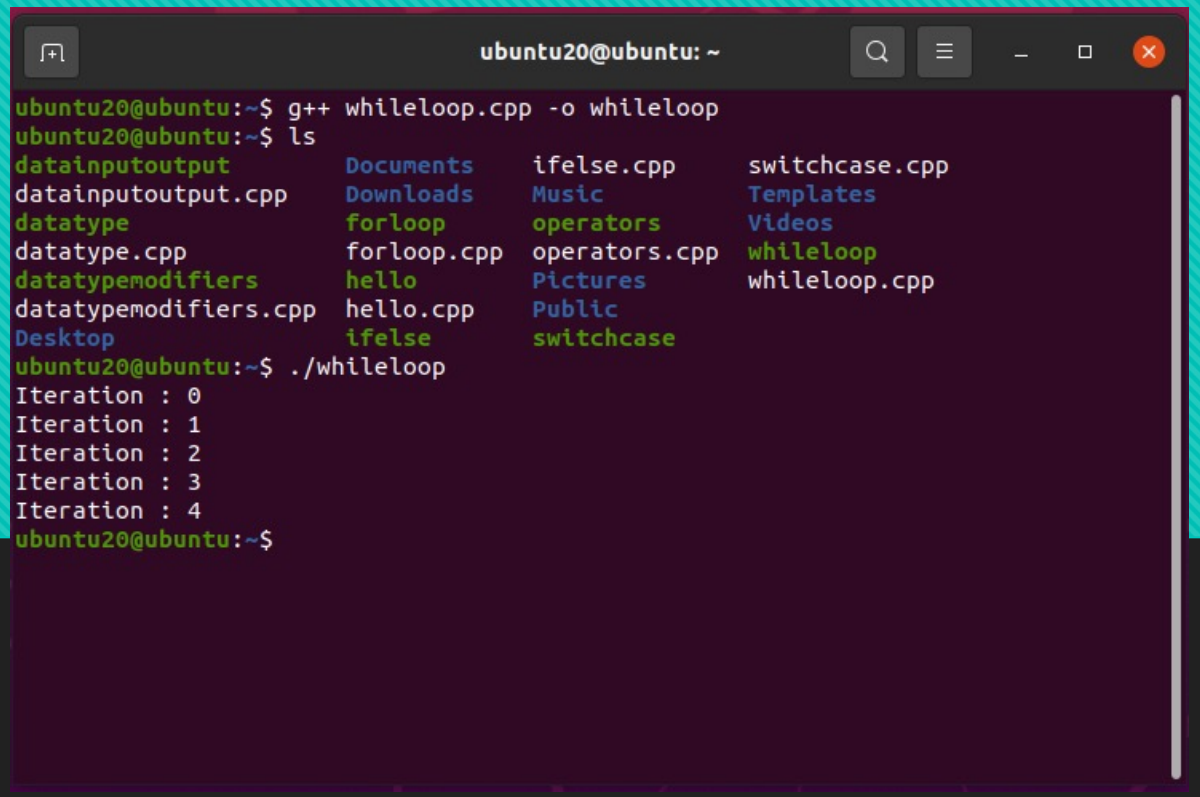
DASL-100.2 C++ Programming and Linux

DASL Drones and Autonomous Systems Lab @ UNLV

5. Loops:

- "while" loop:





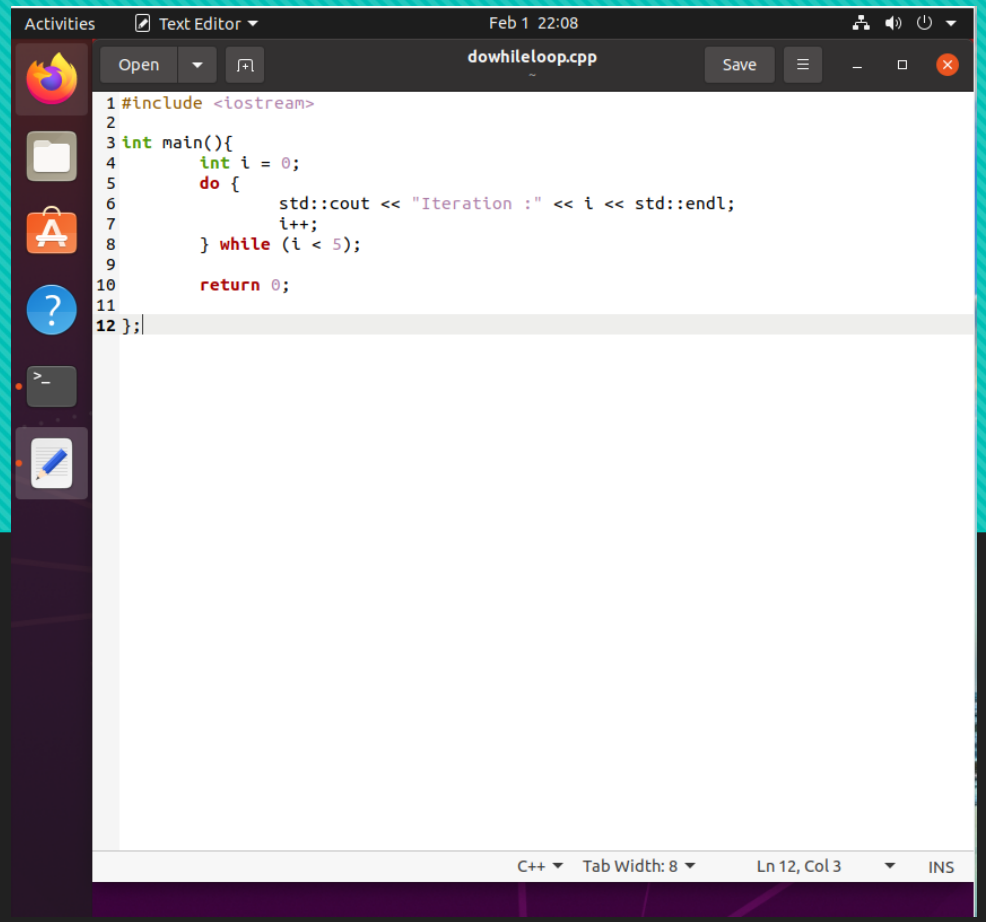
DASL-100.2 C++ Programming and Linux

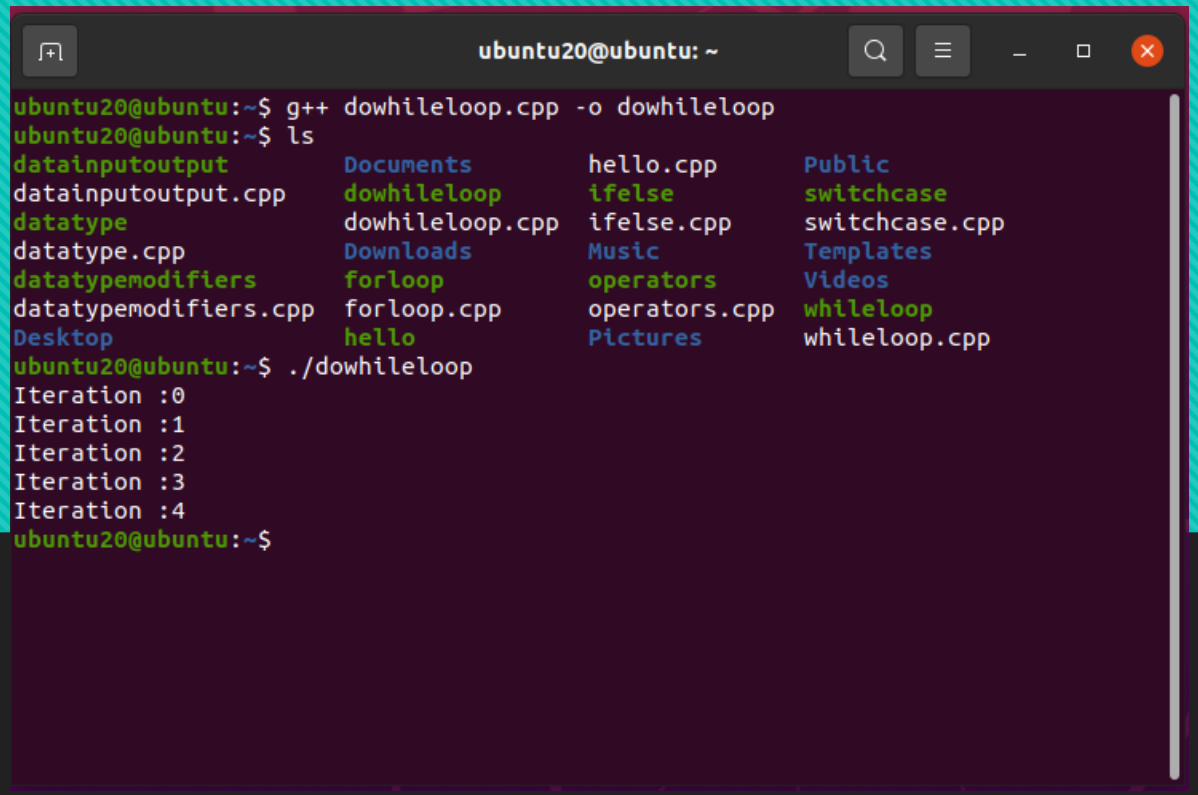
UNLV Howard R. Hughes College of Engineering

DASL Drones and Autonomous Systems Lab @ UNLV

5. Loops:

- "do-while" loop:





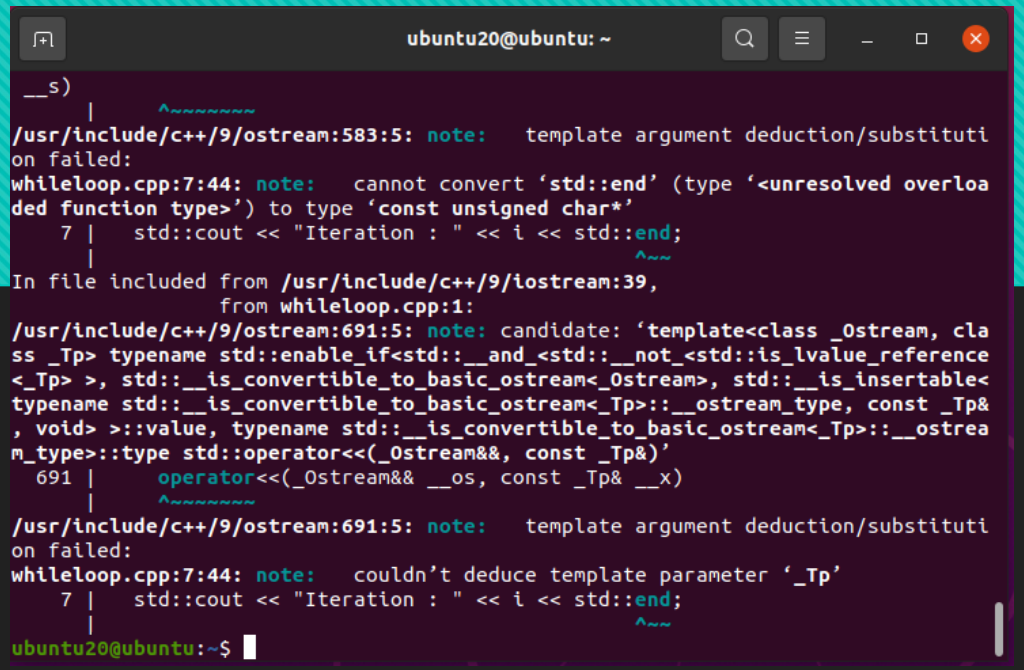
UNLV Howard R. Hughes College of Engineering

DASL-100.2 C++ Programming and Linux

DASL Drones and Autonomous Systems Lab @ UNLV

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.



DASL-100.2
C++ Programming and Linux

Drones and Autonomous Systems Lab @ UNLV

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**.