


§2 C++ Basics

ENGG1111

Computer Programming and Applications

Dirk Schnieders

Outline

- Variables
- Expressions
- Arithmetic Operators
- Relational Operators
- Logical Operators
- Increment and Decrement Operators
- Assignment Operators
- Type Conversion
- Basic Input / Output

Variables

*Once a person has understood the way variables are used in programming,
he has understood the quintessence of programming.*

E. W. DIJKSTRA, *Notes on Structured Programming*

Variables

- Programs manipulate data
 - Such as numbers and letters
- Programming languages use constructs known as variables to name and store data
- Variables are at the very heart of programming
- A C++ variable can hold a number or data of other types
- The content held in a the variable is called its value

Variables

- Variables are like small blackboards
 - We can write a number on them
 - We can change the number
 - We can erase the number
- C++ variables are names for memory locations
 - We can write a value in them
 - We can change the value stored there
 - We cannot erase the memory location
 - Some value is always there

Variables - Declaration

- To use a variable we have to declare it first
- When declaring a variable, we can choose...
 - Variable type
 - Tells the compiler the type of data to store
 - E.g., int is an abbreviation for integer
 - Variable name (the identifier of the variable)
 - Initial value (optional)

```
int width = 5;
```

variable type

variable name (identifier)

initial value

Variables - Declaration

- The data stored in a variable may change over time
 - The computer will assign an appropriate number of memory cells in the main memory to each variable according to the type of data to be stored

```
int width = 5;  
// ...  
width = 6;  
// ...
```

Variable Name (Identifier)

- An identifier must start with either
 - a letter (i.e., A to Z, and a to z), or
 - the underscore symbol (i.e., _)
- The rest of the character may be
 - letters (i.e., A to Z, and a to z),
 - digits (i.e., 0 to 9), or
 - The underscore symbol (i.e., _)
- C++ is case-sensitive
 - So radius, RADIUS, Radius, etc., are different
- Cannot be a keyword in C++

Applications variables3.cpp — ~/cod... 11:00 engg1111

variables3.cpp — ~/code — Atom

variables3.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     int width = 5;
5     int W = 7;
6     int _w = 8;
7 }
8
```

gcc-make-run: Running Command...
"g++" -pedantic-errors -std=c++11 "variables3.cpp" -o "variables3"

✓ gcc-make-run: Build Success





```
variables4.cpp

1 #include <iostream>
2 using namespace std;
3 int main() {
4     int lwidth = 5;
5 }
6
```



gcc-make-run: Running Command...

Close All

```
"g++" -pedantic-errors -std=c++11 "variables4.cpp" -o  
"variables4"
```



gcc-make-run: Compile Error



```
variables4.cpp: In function 'int main()':  
variables4.cpp:4:9: error: expected unqualified-id before  
numeric constant  
    int lwidth = 5;  
           ^~~~~~
```





variables5.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     int c++ = 5;
5 }
6
```

gcc-make-run: Running Command...

Close All

```
"g++" -pedantic-errors -std=c++11 "variables5.cpp" -o  
"variables5"
```

gcc-make-run: Compile Error

x

```
variables5.cpp: In function 'int main()':  
variables5.cpp:4:10: error: expected initializer before  
'++' token  
      int c++ = 5;  
              ^~
```





variables6.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     int __ = 5;
5 }
6
```



gcc-make-run: Running Command...

Close All

```
"g++" -pedantic-errors -std=c++11 "variables6.cpp" -o "variables6"
```



gcc-make-run: Build Success





variables7.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     int r = 7;
5     int R = 9;
6 }
7
```



gcc-make-run: Running Command...

Close All

```
"g++" -pedantic-errors -std=c++11 "variables7.cpp" -o "variables7"
```



gcc-make-run: Build Success





variables8.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     int using = 8;
5 }
6
```



gcc-make-run: Running Command...

Close All

```
"g++" -pedantic-errors -std=c++11 "variables8.cpp" -o "variables8"
```



gcc-make-run: Compile Error



```
variables8.cpp: In function 'int main()':
variables8.cpp:4:7: error: expected unqualified-id before
'using'
    int using = 8;
        ^~~~~~
```





```
variables9.cpp

1 #include <iostream>
2 using namespace std;
3 int main() {
4     int cout = 8; |
5 }
6
```

ⓘ gcc-make-run: Running Command... Close All

```
"g++" -pedantic-errors -std=c++11 "variables9.cpp" -o "variables9"
```

✓ gcc-make-run: Build Success



Keywords in C++

- Reserved words in C++ with predefined meanings
- Cannot be used as names for variables

alignas (since C++11) alignof (since C++11) and and_eq asm atomic_cancel (TM TS) atomic_commit (TM TS) atomic_noexcept (TM TS) auto(1) bitand bitor bool break case catch char char16_t (since C++11) char32_t (since C++11) class(1) compl concept (since C++20) const constexpr (since C++11) const_cast continue decltype (since C++11) default(1) delete(1) do double	dynamic_cast else enum explicit export(1) extern(1) false float for friend goto if import (modules TS) inline(1) int long module (modules TS) mutable(1) namespace new noexcept (since C++11) not not_eq nullptr (since C++11) operator or or_eq private protected public register(2)	reinterpret_cast requires (since C++20) return short signed sizeof(1) static static_assert (since C++11) static_cast struct(1) switch synchronized (TM TS) template this thread_local (since C++11) throw true try typedef typeid typename union unsigned using(1) virtual void volatile wchar_t while xor xor_eq
--	---	---

Variable Name (Identifier)

- Which of the following identifiers are valid in C++?

a_man	2008	program.cc
const	year1-student	_oooo_
an integer	change%2	ABCx123
string	Days_of_week	friend
cout	delete	cos

- Words like cin, cout, string, and cos are NOT keywords in C++
- They are defined in libraries required by the C++ language standard
- Redefining these words, though allowed, can be confusing and thus should be avoided



variables10.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     string string = "test";
5 }
6
```



gcc-make-run: Running Command...

Close All

```
"g++" -pedantic-errors -std=c++11 "variables10.cpp" -o  
"variables10"
```



gcc-make-run: Build Success



Variable Type

- Tells the computer how to interpret the data stored in a variable
- Determines the size of storage needed to store the data
- Some (not all) basic variable types in C++

Name	Description	Size	Range
char	Character or small integer	1 byte	0 to 255 or -128 to 127
bool	Boolean value	1 byte	True(1) or False(0)
int	Integer	4 bytes	-2147483648 to 2147483647
double	Double precision floating point number	8 bytes	~(15 digits)

Variable Type - Declaration

- A declaration specifies a type, and contains a list of one or more variables of that type

```
int age, steps;  
char c;  
bool win;  
double height, width, length;
```

Variable Type - Initialization

- A variable may be initialized in its declaration by using an equal sign followed by a value or an expression

```
int age = 5, steps = age + 10;  
char c = 'Y';  
bool win = true;  
double height = 120.5, length = 1.5e3;
```

- A variable that has not been given a value is said to be uninitialized, and will simply contain some garbage value
- Using uninitialized variables in computations will give unexpected results, and thus should be avoided



variables13.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     int age = 5, steps = age + 10;
5     cout << age << endl;
6     cout << steps << endl;
7     char c = 'Y';
8     cout << c << endl;
9     bool win = true;
10    cout << win << endl;
11    double height = 120.5, length = 1.5e3;
12    cout << height << endl;
13    cout << length << endl;
14 }
```

02/variables13.cpp 14:2

```
X
5
15
Y
1
120.5
1500
Press any key to continue... █
```



Applications variables14.cpp — ~ — ... X variables14

variables14.cpp — Atom

variables14.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     char c = 80;
5     cout << c << endl;
6 }
7
```

X variables14

P
Press any key to continue...■

~/variables14.cpp 6:2



Applications variables14.cpp — ~ — ... X variables15

X variables14.cpp — ~ — Atom

variables15.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     char c = 'P';
5     cout << c << endl;
6 }
7
```

X variables15

P

Press any key to continue... █

~/variables15.cpp 4:14





variables16.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     bool b = true;
5     bool b2 = false;
6     cout << b << endl;
7     cout << b2 << endl;
8 }
9
```

variables14.cpp — ~ — Atom

variables16

X
1
0
Press any key to continue...■





variables17.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     bool b = 0;
5     bool b2 = 1;
6     bool b3 = -2;
7     cout << b << endl;
8     cout << b2 << endl;
9     cout << b3 << endl;
10 }
11
```





variables18.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     bool b = true;
5     bool b2 = false;
6     bool b3 = true;
7     b = b2;
8     b2 = b3;
9     b3 = false;
10    cout << b << endl;
11    cout << b2 << endl;
12    cout << b3 << endl;
13 }
14
```





variables19.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     double d = 1.01;
5     double d2 = 1.01e5;
6     cout << d << endl;
7     cout << d2 << endl;
8 }
9
```

X variables19
1.01
101000
Press any key to continue... █





variables20.cpp

•

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     int i = -2147483648;
5     int i2 = 2147483647;
6     cout << i;
7 }
8 |
```





variables21.cpp

```
1 //Declaration
2 #include <iostream>
3 using namespace std;
4 int main () {
5     int a, b, c;
6     cout << a << " " << b << " " << c << endl;
7 }
8
```



Initializations: Constants

- Sometimes we want to assign a fixed value to a variable

```
double PI = 3.14159265359;
```

- const modifier
 - Add in front of a variable declaration to declare the variable as a constant
 - Compiler will make sure that the variable remains a constant



File Edit View Selection Find Packages Help

variables22.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     const double PI = 3.14159265359;
5     PI = 42;
6 }
7
```



gcc-make-run: Running Command...

Close All

```
"g++" -pedantic-errors -std=c++11 "variables22.cpp" -o "variables22"
```



gcc-make-run: Compile Error

X

```
variables22.cpp: In function 'int main()':
variables22.cpp:5:7: error: assignment of read-only
variable 'PI'
    PI = 42;
          ^~
```



Expressions

Expressions

- Combine variables and constants to produce new values
- Composed of operands and operators
- Operand
 - Data on which the computation is performed
 - May be variable or constant
- Operator
 - Specifies what is to be done on the operands
 - E.g., Arithmetic operators, relational operators, logical operators

```
radius * 2 * 3.1416
```



File Edit View Selection Find Packages Help

variables23.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     const double PI = 3.14159265359;
5     double radius = 2.0;
6     double c = 2 * PI * radius;
7     cout << PI << endl;
8     cout << radius << endl;
9     cout << c << endl;
10 }
11
```

T

variables23

3.14159

2

12.5664

Press any key to continue... █



File Edit View Selection Find Packages Help

variables24.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     const double PI = 3.14159265359;
5     double radius = 2.0;
6     double c = 2 * PI * radius;
7     cout.precision(20);
8     cout << PI << endl;
9     cout << radius << endl;
10    cout << c << endl;
11 }
12
```

T

variables24

3.1415926535900000616

2

12.566370614360000246

Press any key to continue... █



Operators

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Increment and Decrement Operators
- Assignment Operators

Arithmetic Operators

Arithmetic Operators

Arithmetic Operators	Sign in the expression
Addition	+
Subtraction	-
Multiplication	*
Division	/
Modulus	%

- Modulus operator produces the remainder
 - E.g., $10 \% 3$ equals 1
 - $10 = (3 * 3) + 1$



File Edit View Selection Find Packages Help

arithmeticOperations3.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     int a = 3;
5     int b = 2;
6     cout << a + b << endl;
7     cout << a - b << endl;
8     cout << a * b << endl;
9     cout << a / b << endl;
10    cout << a % b << endl;
11 }
12
```

X

arithmeticOperations3

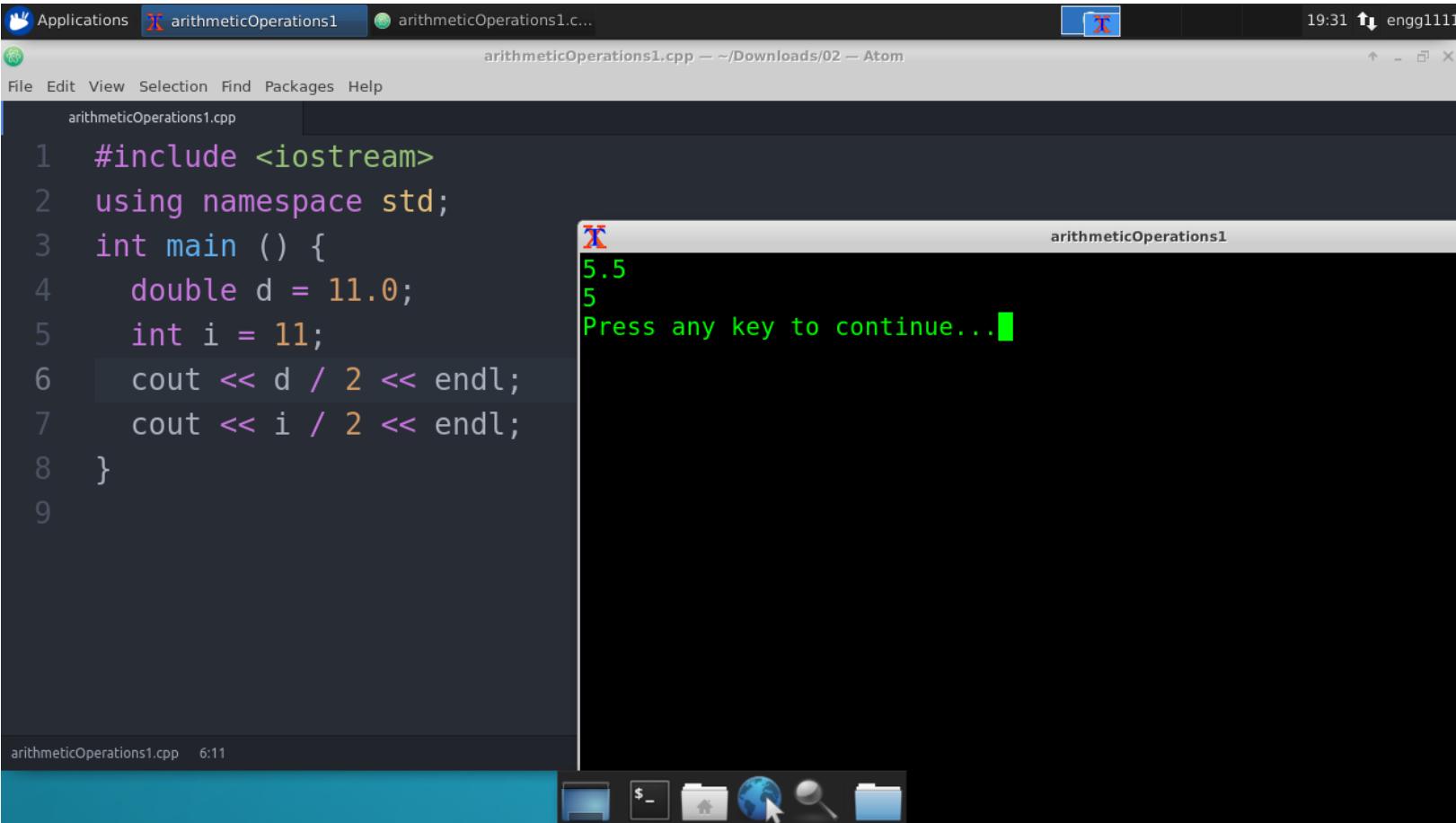
5
1
6
1
1

Press any key to continue... █



Arithmetic Operators

- Note that integer division truncates any fractional part



The screenshot shows a desktop environment with a window titled "arithmeticOperations1" in the Atom code editor. The code in the editor is:

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     double d = 11.0;
5     int i = 11;
6     cout << d / 2 << endl;
7     cout << i / 2 << endl;
8 }
```

To the right of the editor is a terminal window titled "arithmeticOperations1" showing the output of the program:

```
5.5
5
Press any key to continue... █
```

The bottom of the terminal window shows the status bar with "arithmeticOperations1.cpp 6:11".

Arithmetic Operators

- The % operator cannot be applied to double

The screenshot shows a terminal window within the Atom code editor. The terminal output is as follows:

```
gcc-make-run: Running Command...
g++ -pedantic-errors -std=c++11
"arithmeticOperations2.cpp" -o "arithmeticOperations2"
gcc-make-run: Compile Error
arithmeticOperations2.cpp: In function 'int main()':
arithmeticOperations2.cpp:6:12: error: invalid operands
of types 'double' and 'int' to binary 'operator%'
cout << d % 2 << endl;
^~~~~~
```

The code in the editor is:

```
#include <iostream>
using namespace std;
int main () {
    double d = 11.0;
    int i = 11;
    cout << d % 2 << endl;
    cout << i % 2 << endl;
}
```

The status bar at the bottom indicates the file is 5:13 and there are 0 files.



File Edit View Selection Find Packages Help

arithmeticOperations4.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     cout << 1 + 2 - 3 * 4 / 5 % 6 << endl;
5 }
6 |
```



Precedence

- In evaluating an expression with mixed operators, those operators with a higher priority will be carried out before those with a lower priority

```
1 + 2 * 3
```

- The operator * has a higher precedence than the operator +
 - The order of evaluation is equivalent to $1 + (2 * 3)$ and the result is 7

Precedence

- In evaluating an expression with mixed operators, those operators with a higher priority will be carried out before those with a lower priority

$$12 - 11 \% 3$$

- The operator % has a higher precedence than the operator -
 - The order of evaluation is equivalent to $12 - (11 \% 3)$ and the result is 10

Associativity

- Associativity rule is used to determine the order of evaluation for operators with the same precedence in an expression

2 * 3 % 2

- The operators * and % have the same precedence
 - The order of evaluation is equivalent to $(2 * 3) \% 2$, and the result is 0

Precedence & Associativity

Operators	Associativity
!	-
* , / , %	left to right
+ , -	left to right
< , <= , > , >=	left to right
== , !=	left to right
&&	left to right
	left to right
= , += , -= , *= , /= , %=	right to left

High precedence



Lower precedence

- The order of evaluation may be overridden by inserting parentheses () into the expressions
 - E.g., $(1 + 2)^* 3$

Relational Operators

Relational Operators

Relational Operators	Sign in the expression
Greater than	>
Greater than or equal	\geq
Smaller than	<
Smaller than or equal	\leq
Equal	\equiv
Not equal	\neq

- Usually used with the if-then-else or the while loop to control the flow of the program (more in §3)



File Edit View Selection Find Packages Help

relationalOperators1.cpp

```
2 using namespace std;
3 int main () {
4     int a = 2, b = 2;
5     cout << (a > b) << endl;
6     cout << (a >= b) << endl;
7     cout << (a < b) << endl;
8     cout << (a <= b) << endl;
9     cout << (a == b) << endl;
10    cout << (a != b) << endl;
11 }
12
```

relationalOperators1

0
1
0
1
1
0

Press any key to continue... █



Relational Operators

- In C++, the numeric value of a relational or logical expression is 1 if the relation is true, and 0 if the relation is false

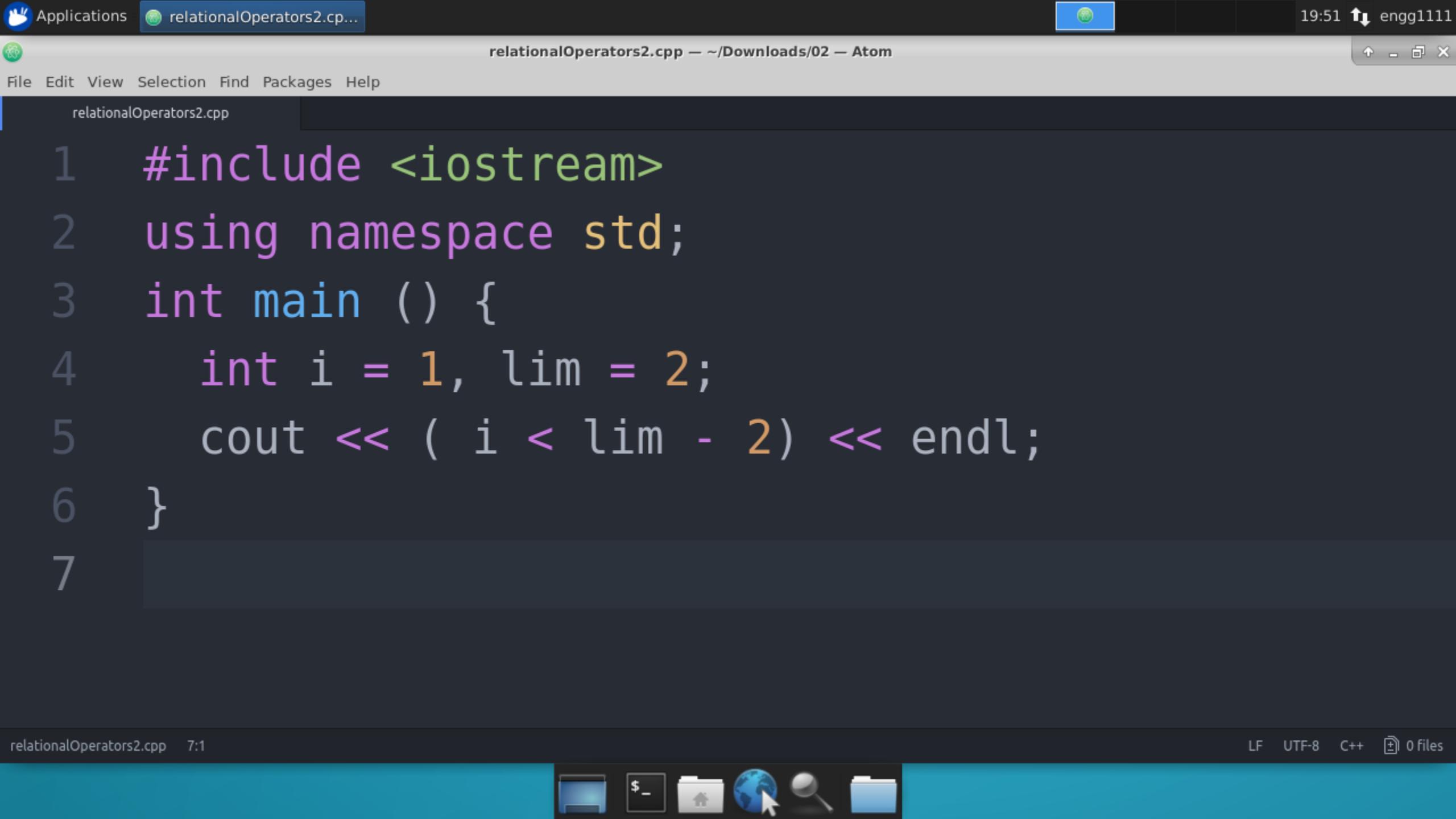
Applications relationalOperators2.cpp... 19:51 engg1111

File Edit View Selection Find Packages Help

relationalOperators2.cpp — ~/Downloads/02 — Atom

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     int i = 1, lim = 2;
5     cout << ( i < lim - 2) << endl;
6 }
7
```

relationalOperators2.cpp 7:1 LF UTF-8 C++ 0 files



File Edit View Selection Find Packages Help

relationalOperators3.cpp — ~/Downloads/02 — Atom



```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     int i = 1, lim = 2;
5     cout << ( (i < lim) - 2) << endl;
6 }
7
```



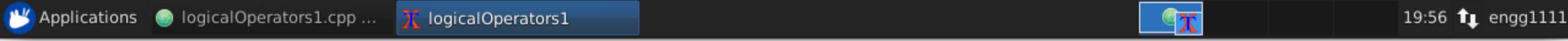
Logical Operators

Logical Operators

Logical Operators	Sign in the expression
And	<code>&&</code>
Or	<code> </code>
Not	<code>!</code>

A	B	A && B	A B
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

- The unary negation operator `!` converts a non-zero operand into 0, and a zero operand into 1



logicalOperators1.cpp — ~/Downloads/02 — Atom

File Edit View Selection Find Packages Help

logicalOperators1.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     bool test = !0;
5     cout << test << endl;
6 }
7
```

logicalOperators1.cpp 5:23

Atom interface elements: a blue square with a white 'X', a dollar sign icon, a house icon, a globe icon, a magnifying glass icon, and a folder icon.

logicalOperators1

```
1
Press any key to continue... █
```

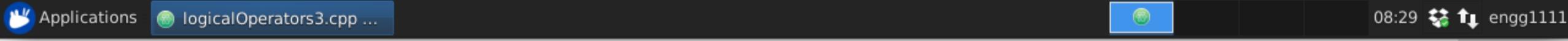


File Edit View Selection Find Packages Help

logicalOperators2.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     bool test = !0 && 0;
5     cout << test << endl;
6 }
7
```





logicalOperators3.cpp — ~/Dropbox/work/teaching/17-18 engg1111/VM/02 — Atom

File Edit View Selection Find Packages Help

logicalOperators3.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     bool b;
5     bool test = !b && b;
6     cout << test << endl;
7 }
8
```

logicalOperators3.cpp 6:14

LF UTF-8 C++ 0 files





File Edit View Selection Find Packages Help

logicalOperators4.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     bool test = true || false && true;
5     test = !test;
6     cout << test << endl;
7 }
8
```



Increment and Decrement Operators

Increment and Decrement

Assignment Operators	Sign in the expression
Increment	<code>++</code>
Decrement	<code>--</code>

- The increment operator `++` adds 1 to its operand
 - I.e., the statement `i++` is equivalent to `i=i+1`
- The decrement operator `--` subtracts 1 from its operand
 - I.e., the statement `i--` is equivalent to `i=i-1`



File Edit View Selection Find Packages Help

```
incrementAndDecrement1.cpp
```

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     int x = 0;
5     x++;
6     cout << x << endl;
7 }
8
```

```
incrementAndDecrement1
1
Press any key to continue...
```



Increment and Decrement

- The operators `++` and `--` may be used either as prefix (e.g., `++i`) or postfix (e.g., `i++`) operators
 - When used as prefix, the increment/decrement is done before the value is used
 - When used as postfix, the increment/decrement is done after the value is used

File Edit View Selection Find Packages Help

incrementAndDecrement2.cpp — ~/Downloads/02 — Atom

incrementAndDecrement2.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     int x = 0;
5     cout << ++x << endl;
6     cout << x << endl;
7 }
8
```



File Edit View Selection Find Packages Help

incrementAndDecrement3.cpp — ~/Downloads/02 — Atom

incrementAndDecrement3.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     int x = 0;
5     cout << x++ << endl;
6     cout << x << endl;
7 }
8
```



File Edit View Selection Find Packages Help

incrementAndDecrement4.cpp — ~/Downloads/02 — Atom

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     int x = 0;
5     cout << (false&&x++) << endl;
6     cout << x << endl;
7 }
8
```



Assignment Operators

Assignment Operators

Assignment Operators	Sign in the expression
Assignment	=
Addition	+=
Subtraction	-=
Multiplication	*=
Division	/=

- Expression such as $i=i+2$ in which the variable on the left hand side is repeated immediately on the right can be written in the compressed form $i+=2$



File Edit View Selection Find Packages Help

assignmentOperators1.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     int x = 7;
5     x = x+10;
6     cout << x << endl;
7 }
8
```

X assignmentOperators1
17
Press any key to continue... █





File Edit View Selection Find Packages Help

```
1 #include <iostream>
2 using namespace std;
3 int main () {
4     int x = 7;
5     x+=10;
6     cout << x << endl;
7 }
8
```

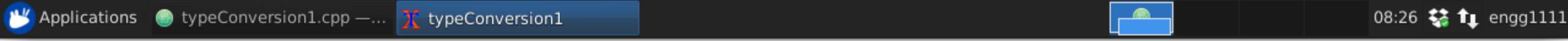
X assignmentOperators2
17
Press any key to continue... █

Type Conversion

Task

- Write a program to that converts degree Celsius to Fahrenheit

$$f = \frac{9}{5}c + 32$$



typeConversion1.cpp — ~/Dropbox/work/teaching/17-18 engg1111/VM/02 — Atom

↑ _ □ ×

File Edit View Selection Find Packages Help

typeConversion1.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     int c = 1;
5     cout << c << "c = " << 9 / 5 * c + 32 << "f" << endl;
6 }
7
```

X typeConversion1
1c = 33f
Press any key to continue... █

typeConversion1.cpp 5:46

0 files





File Edit View Selection Find Packages Help

typeConversion1_1.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     int c = 1;
5     cout << c << "c = " << 9 / 5.0 * c + 32 << "f" << endl;
6 }
7
```

X typeConversion1_1

```
1c = 33.8f
Press any key to continue... █
```

typeConversion1_1.cpp 5:32 0 files



Applications typeConversion2.cpp —... typeConversion2

typeConversion2.cpp — ~/Downloads/02 — Atom

File Edit View Selection Find Packages Help

typeConversion2.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     int c = 1.99;
5     cout << c << endl;
6 }
7
```

X typeConversion2

1
Press any key to continue... █

typeConversion2.cpp 4:14

files

The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "typeConversion2" and it contains the output of the program "typeConversion2.cpp". The output is "1" followed by a message "Press any key to continue...". The desktop interface includes a dock at the bottom with icons for terminal, file manager, and system tools.



File Edit View Selection Find Packages Help

typeConversion3.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     double d = 1.99;
5     int c = d;
6     cout << c << endl;
7 }
```

typeConversion3

```
1
Press any key to continue... █
```

typeConversion3.cpp 5:11



File Edit View Selection Find Packages Help

typeConversion4.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     double d = 3000000000;
5     int c = d;
6     cout << c << endl;
7 }
```

-2147483648
Press any key to continue... █

typeConversion4.cpp 4:23

Type Conversion

- In arithmetic expressions, if a binary operator has operands of different types, the “lower” type is promoted to the “higher” type before the operation proceeds and the result is of the higher type

Applications typeConversion5.cpp —... typeConversion5

typeConversion5.cpp — ~/Downloads/02 — Atom

File Edit View Selection Find Packages Help

typeConversion5.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     cout << 3 / 2 << endl;
5     cout << 3.0 / 2 << endl;
6 }
7
```

X typeConversion5

1
1.5
Press any key to continue... █

typeConversion5.cpp 5:1

0 files

The screenshot shows a Linux desktop environment with several windows open. At the top, there's a dock with icons for Applications, Home, and a file manager. The title bar of the active window says "typeConversion5.cpp — ~/Downloads/02 — Atom". The menu bar includes File, Edit, View, Selection, Find, Packages, and Help. Below the menu is a tab bar with "typeConversion5.cpp". The main area contains C++ code with line numbers 1 through 7. Lines 1-6 are part of a program that prints 1 and 1.5 to the console. Line 7 is a prompt for the user to press a key. A terminal window titled "typeConversion5" is visible, showing the output of the program: "1", "1.5", and "Press any key to continue... █". The bottom of the screen shows the desktop environment's taskbar with icons for Home, Terminal, File Manager, and a search bar.

Basic Input / Output

Basic I/O

- C++ uses a convenient abstraction called streams to perform input and output operations in sequential media such as the screen or the keyboard
- A stream is an object where a program can either insert or extract characters to/from
- The standard C++ library includes the header file `iostream` where the standard input and output stream objects are declared

Output (cout)

- By default, the standard output of a program is the screen, and the C++ stream object defined to access it is cout
- The insertion operator << is used to insert data into the stream
- The insertion operation may be used more than once in a single statement



File Edit View Selection Find Packages Help

basicInputOutput1.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     int a=1, b=2;
5     cout << "a=" << a << " b=" << b << endl;
6 }
7
```

X basicInputOutput1

```
a=1 b=2
Press any key to continue... █
```

basicInputOutput1.cpp 5:23 0 files

Output (cout)

- There are some escape sequences that have special usage in the output

\a	alert (bell) character	\v	vertical tab
\b	backspace	\\"	backslash
\n	newline	\?	question mark
\r	carriage return	\'	single quote
\t	horizontal tab	\"	double quote

Input (cin)

- The standard input device is usually the keyboard, and the C++ stream object defined to access it is cin
- The extraction operator >> is used to extract data from the stream



File Edit View Selection Find Packages Help

basicInputOutput2.cpp

basicInputOutput2.cpp — ~/Downloads/02 — Atom

↑ - X

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     int age, height, weight;
5     cin >> age >> height >> weight;
6     cout << "age=" << age << endl;
7     cout << "height=" << height << endl;
8     cout << "weight=" << weight << endl;
9 }
10
```

basicInputOutput2.cpp 9:2

LF UTF-8 C++ 0 files



Input (cin)

- The type of the variable will determine the type of data that is extracted from the stream
- Note that cin can only process the input from the keyboard once the RETURN key has been pressed