§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;
```

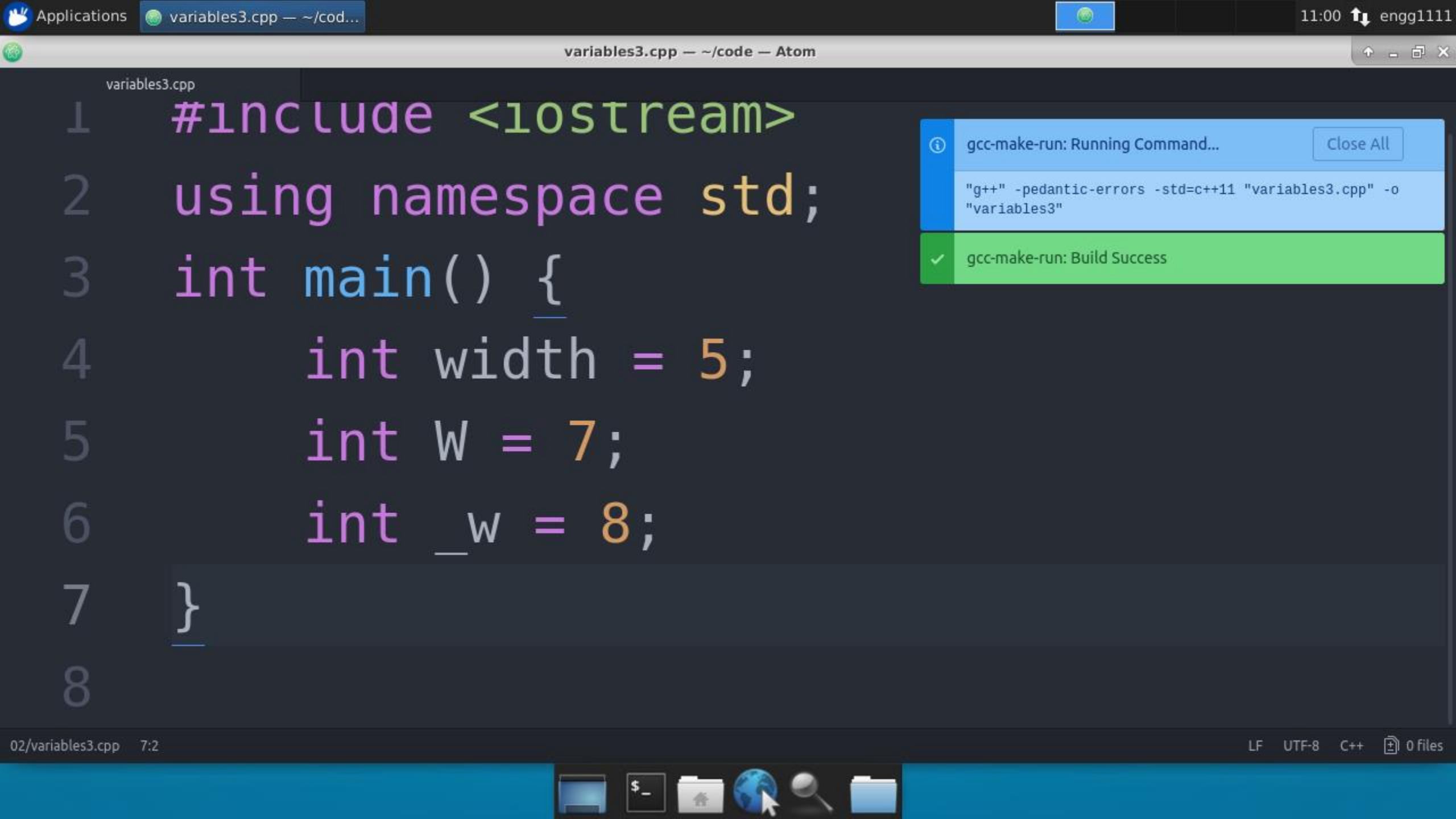
Variables - Declaration

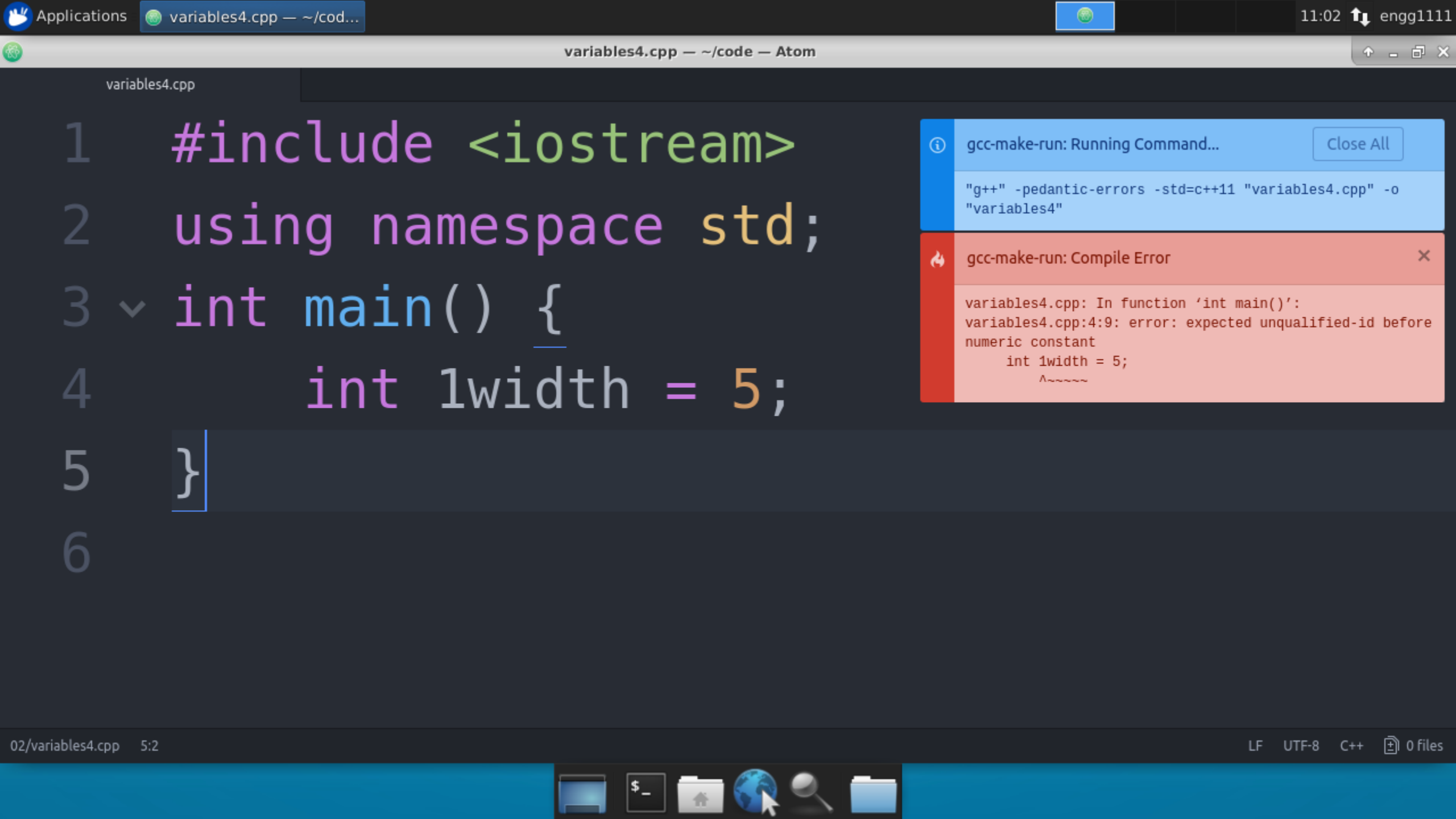
- The data stored in a variable may change over time
  - The computer will assign an appropriate number of memory cells in the main memory to each variable according to the type of data to be stored

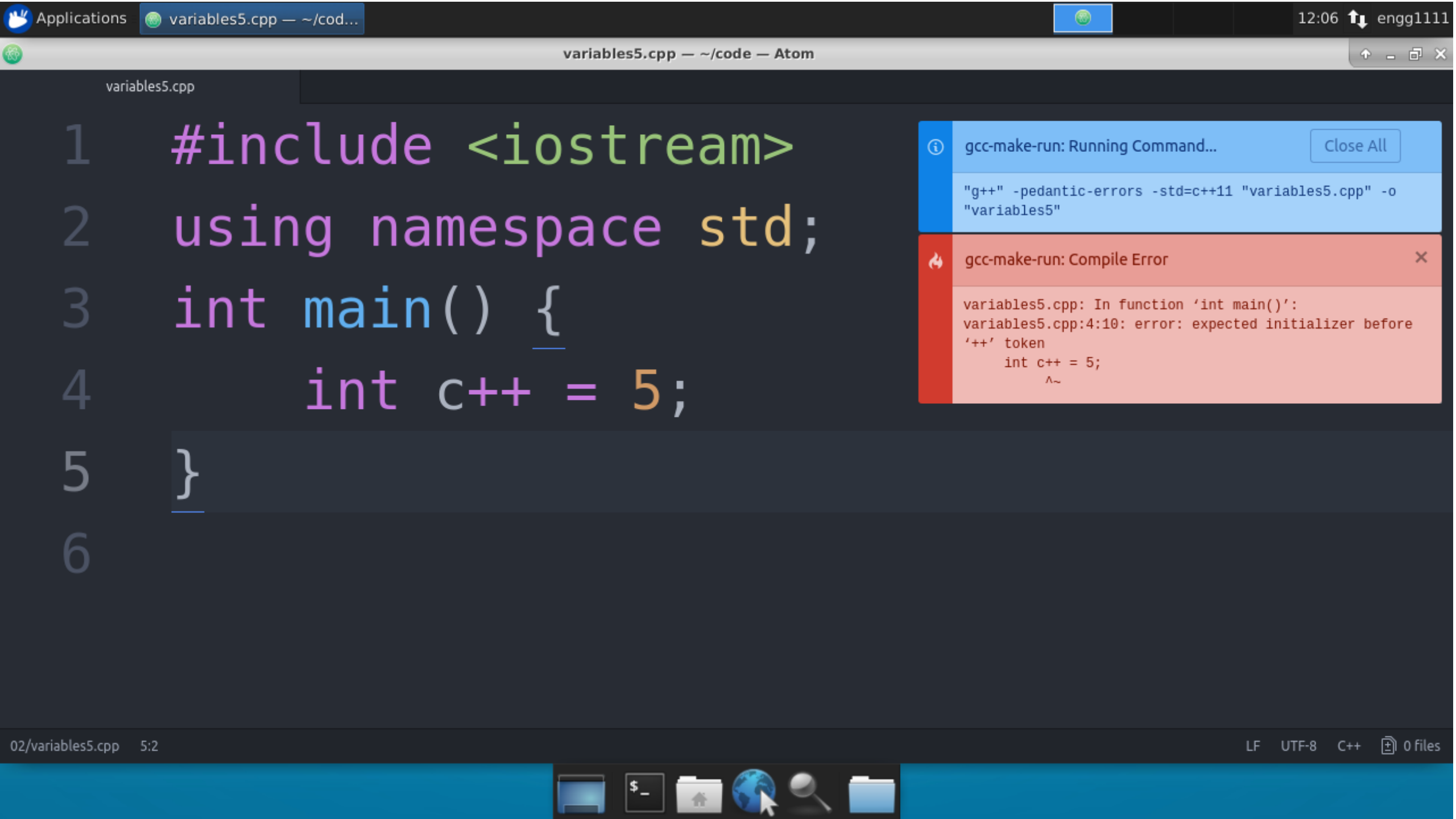
```
int width = 5;
// ...
width = 6;
// ...
```

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
#include <iostream>
using namespace std;
int main() {
int width = 5;
int W = 7;
int _w = 8;
}
gcc-make-run: Running Command...
Close All
"g++" -pedantic-errors -std=c++11 "variables3.cpp" -o "variables3"
gcc-make-run: Build Success
02/variables3.cpp 7:2
LF UTF-8 C++ 0 files


Applications
variables4.cpp — ~/cod...
11:02
engg1111
variables4.cpp — ~/code — Atom
variables4.cpp
1 #include <iostream>
2 using namespace std;
3 int main() {
4 int 1width = 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 1width = 5;
^~~~~~
02/variables4.cpp 5:2
LF UTF-8 C++ 0 files


Applications
variables5.cpp — ~/cod...
12:06
engg1111
variables5.cpp — ~/code — Atom
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
variables5.cpp: In function ‘int main()’:
variables5.cpp:4:10: error: expected initializer before ‘++’ token
int c++ = 5;
^~
02/variables5.cpp 5:2
LF UTF-8 C++ 0 files

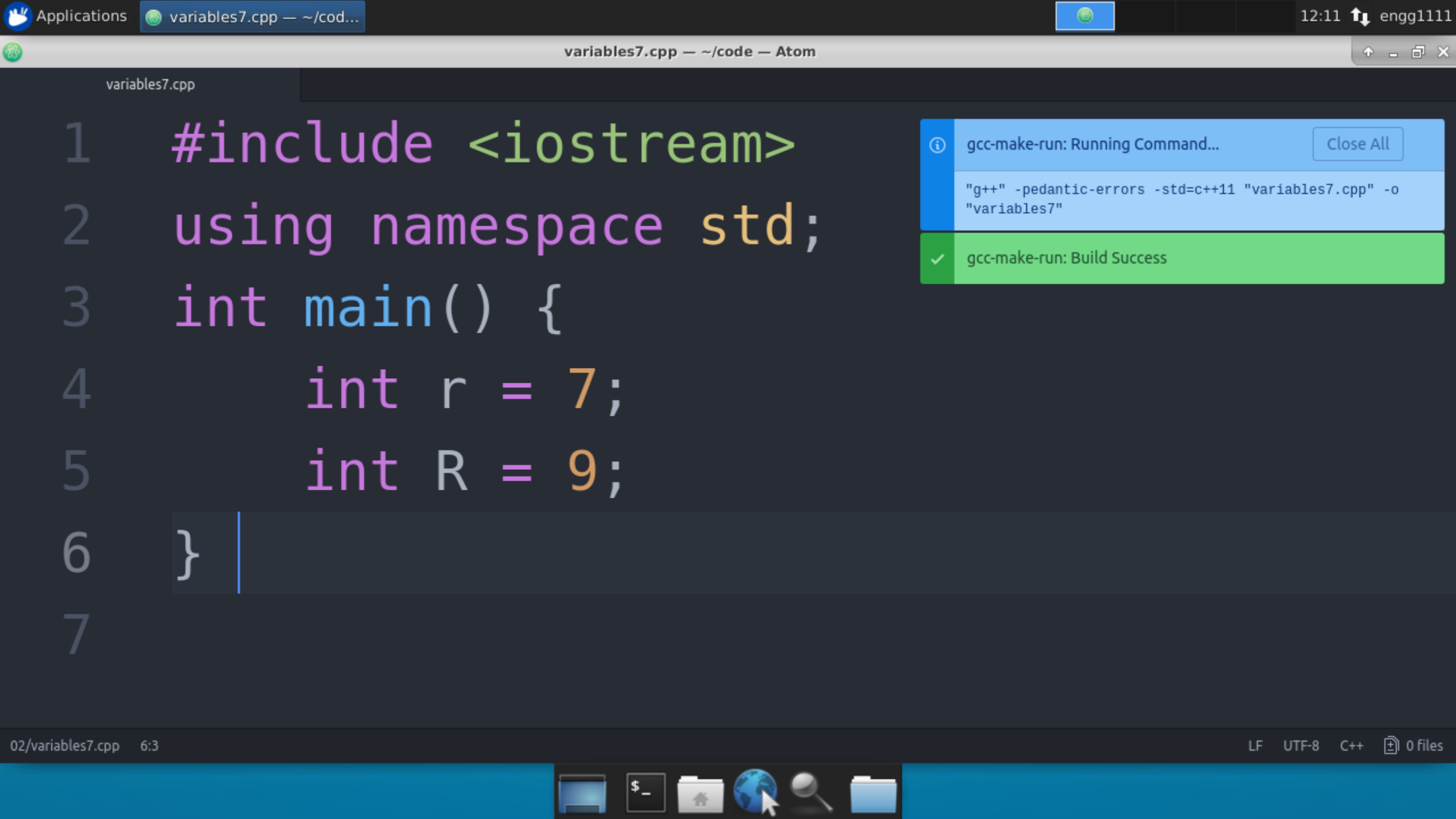
variables6.cpp

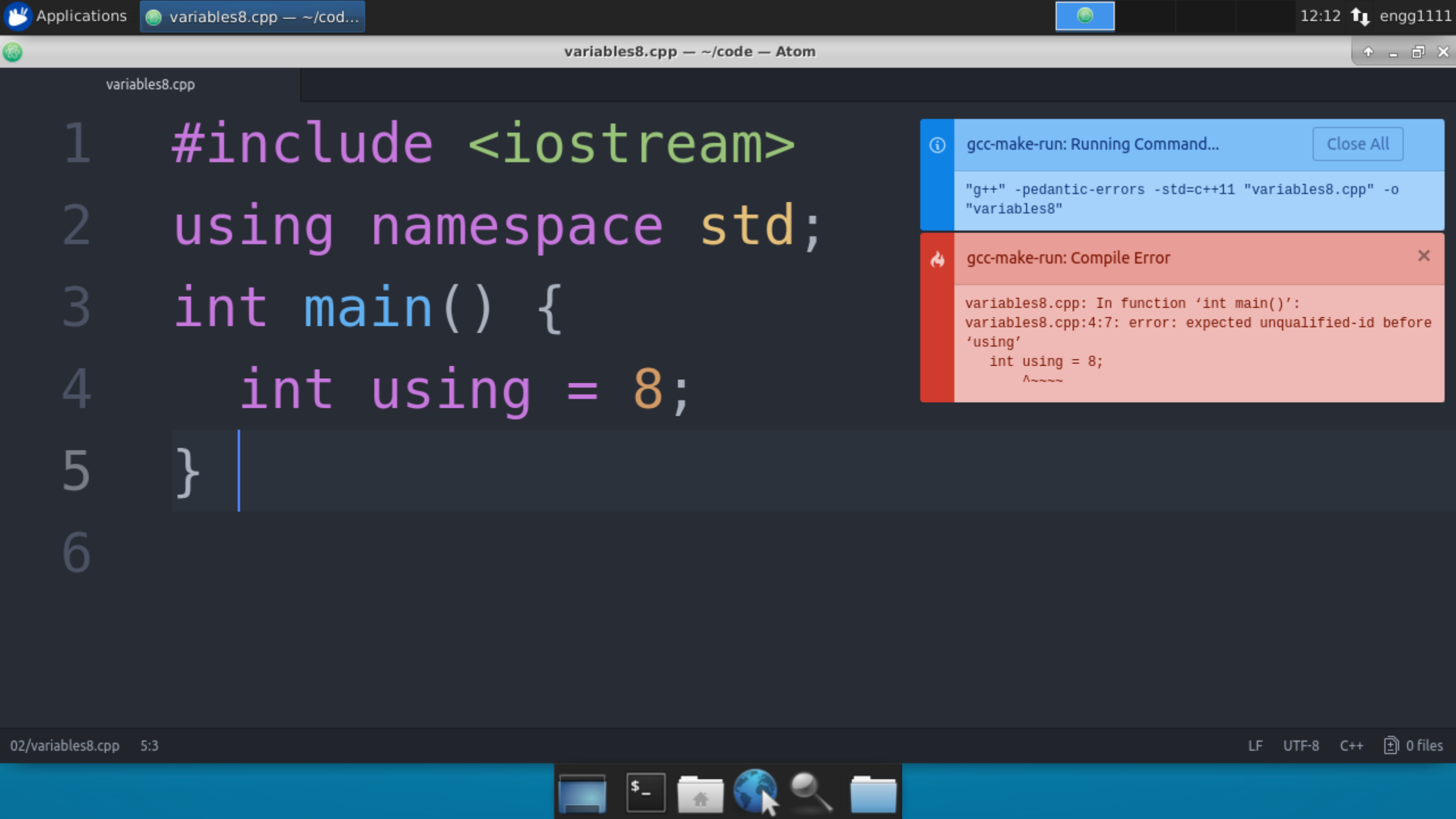
```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int __ = 5;
5  }
6
```

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

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

gcc-make-run: Build Success


Applications
variables7.cpp — ~/cod...
12:11
engg1111
variables7.cpp — ~/code — Atom
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
02/variables7.cpp 6:3
LF UTF-8 C++ 0 files


Applications
variables8.cpp — ~/cod...
12:12
engg1111
variables8.cpp — ~/code — Atom
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;
^~~~~
02/variables8.cpp 5:3
LF UTF-8 C++ 0 files

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	-2147483648to 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

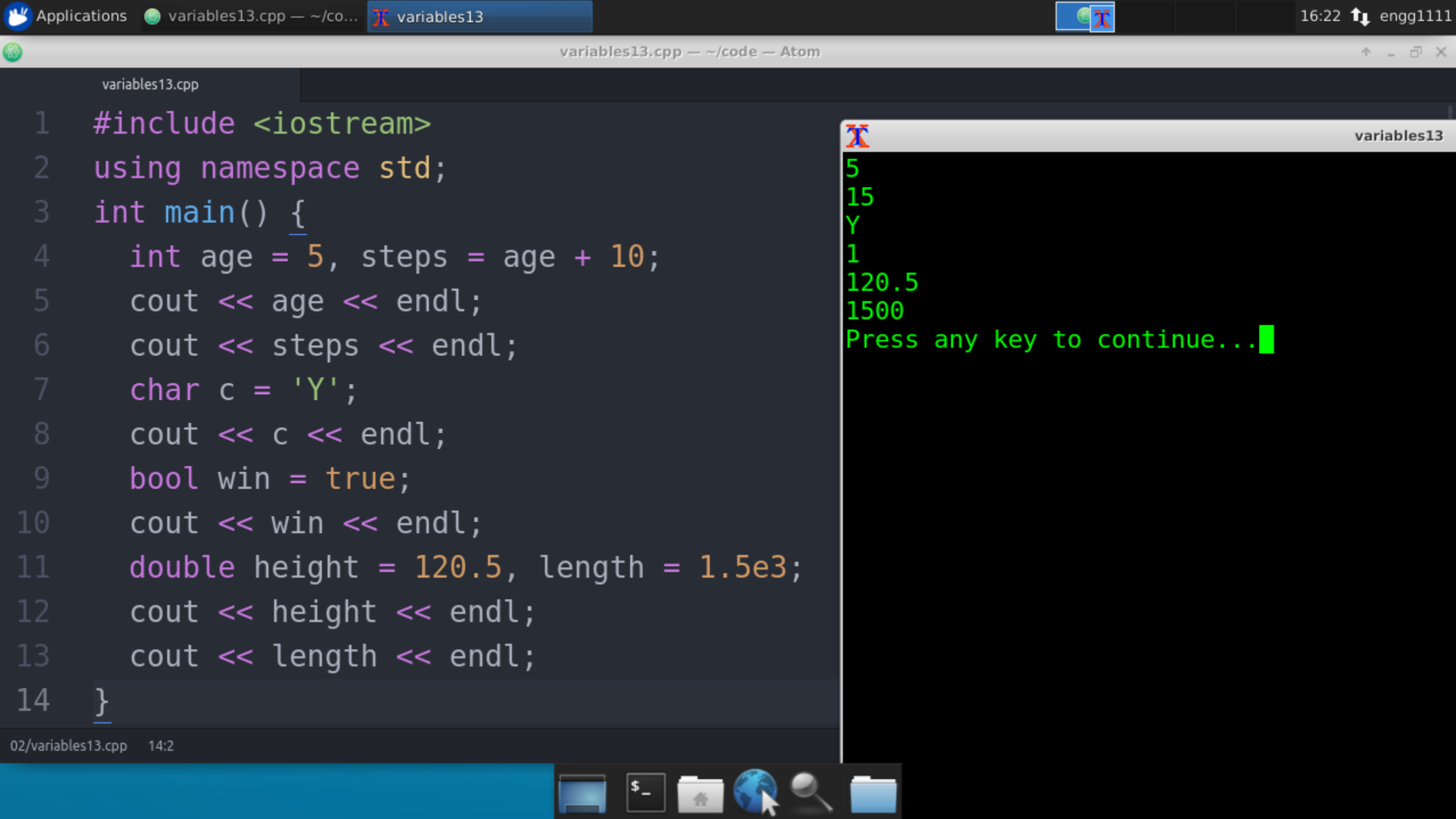
```
int age, steps;
char c;
bool win;
double height, width, length;
```

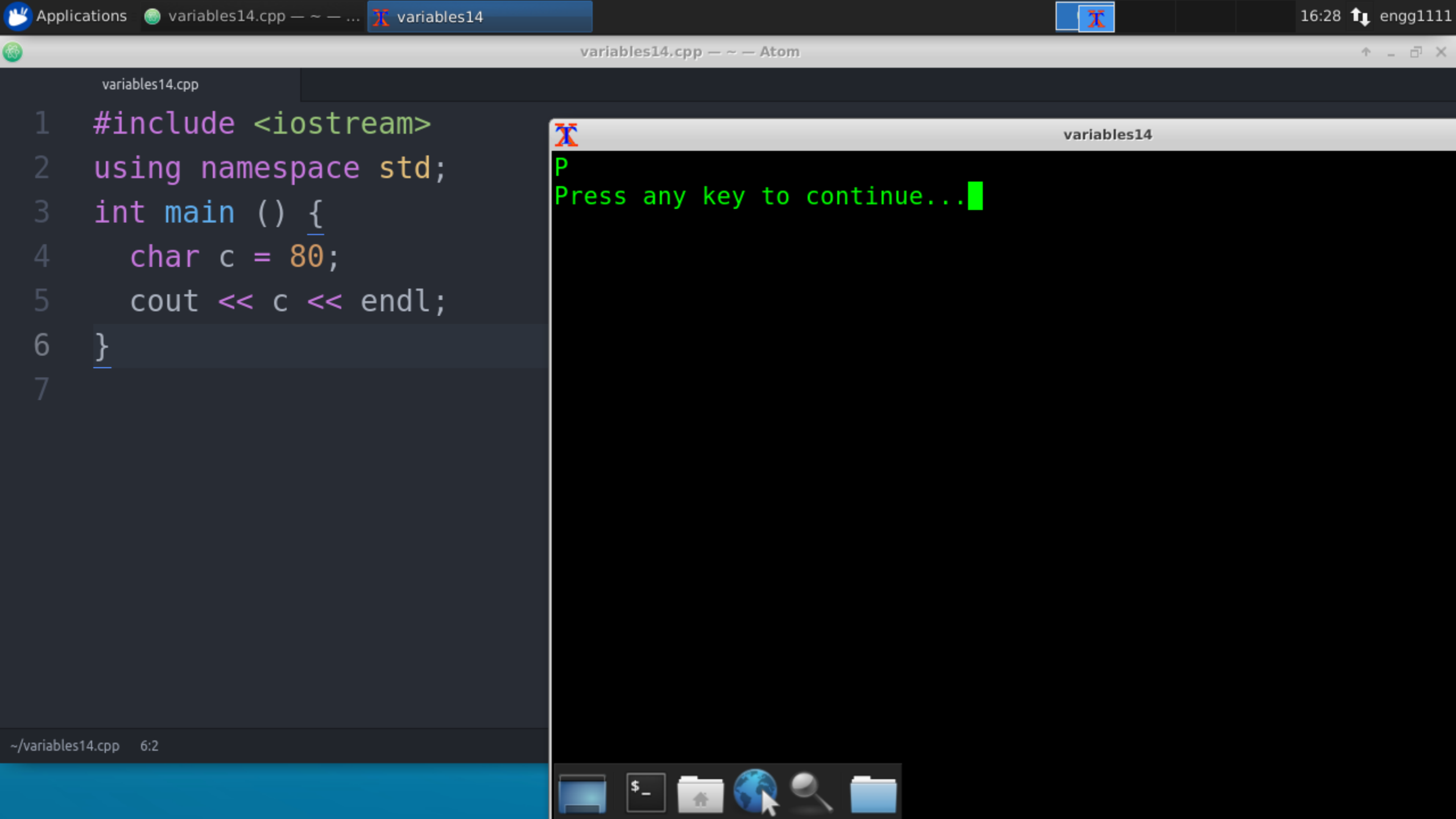
Variable Type - Initialization

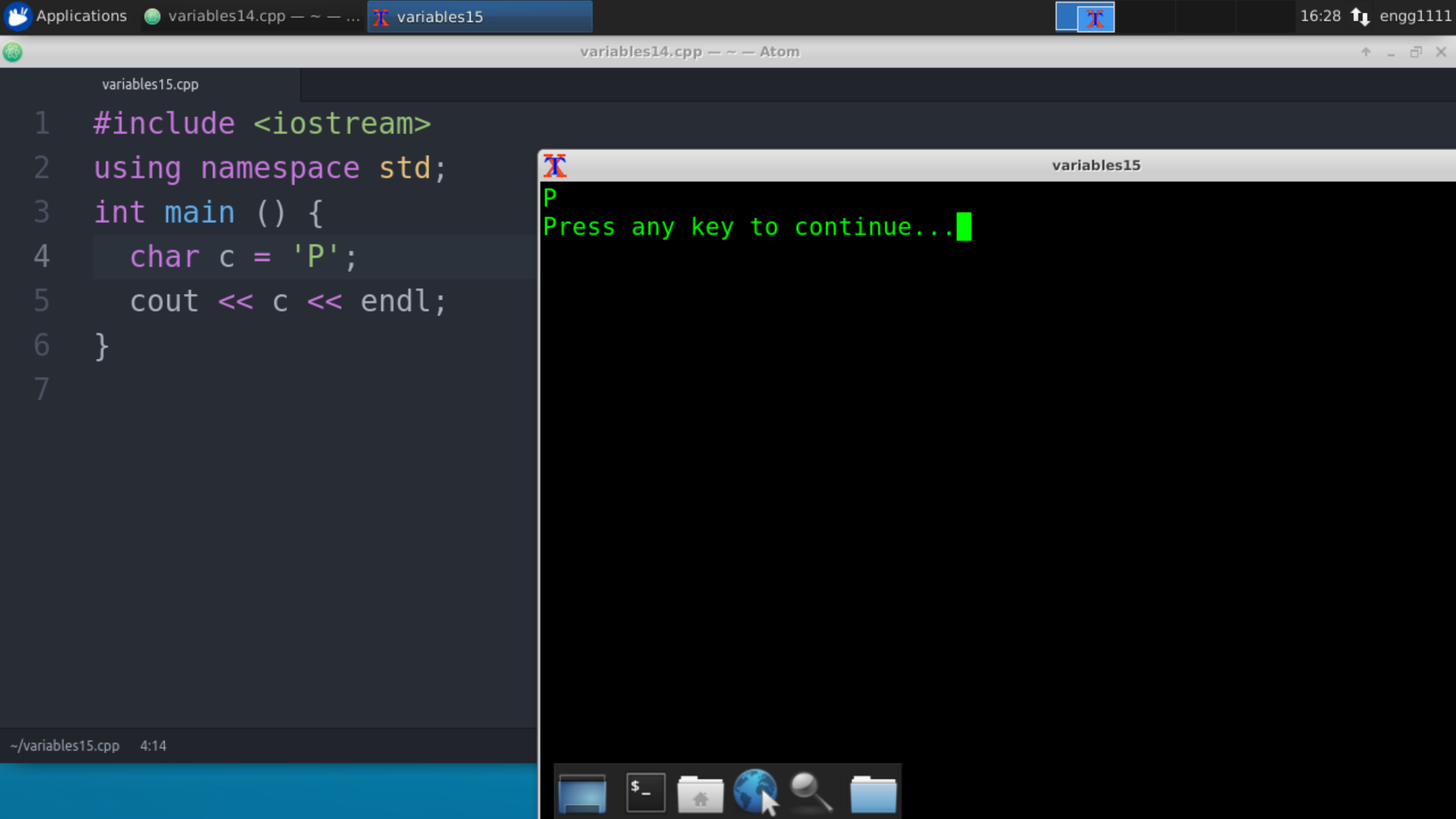
- A variable may be initialized in its declaration by using an equal sign followed by a value or an expression

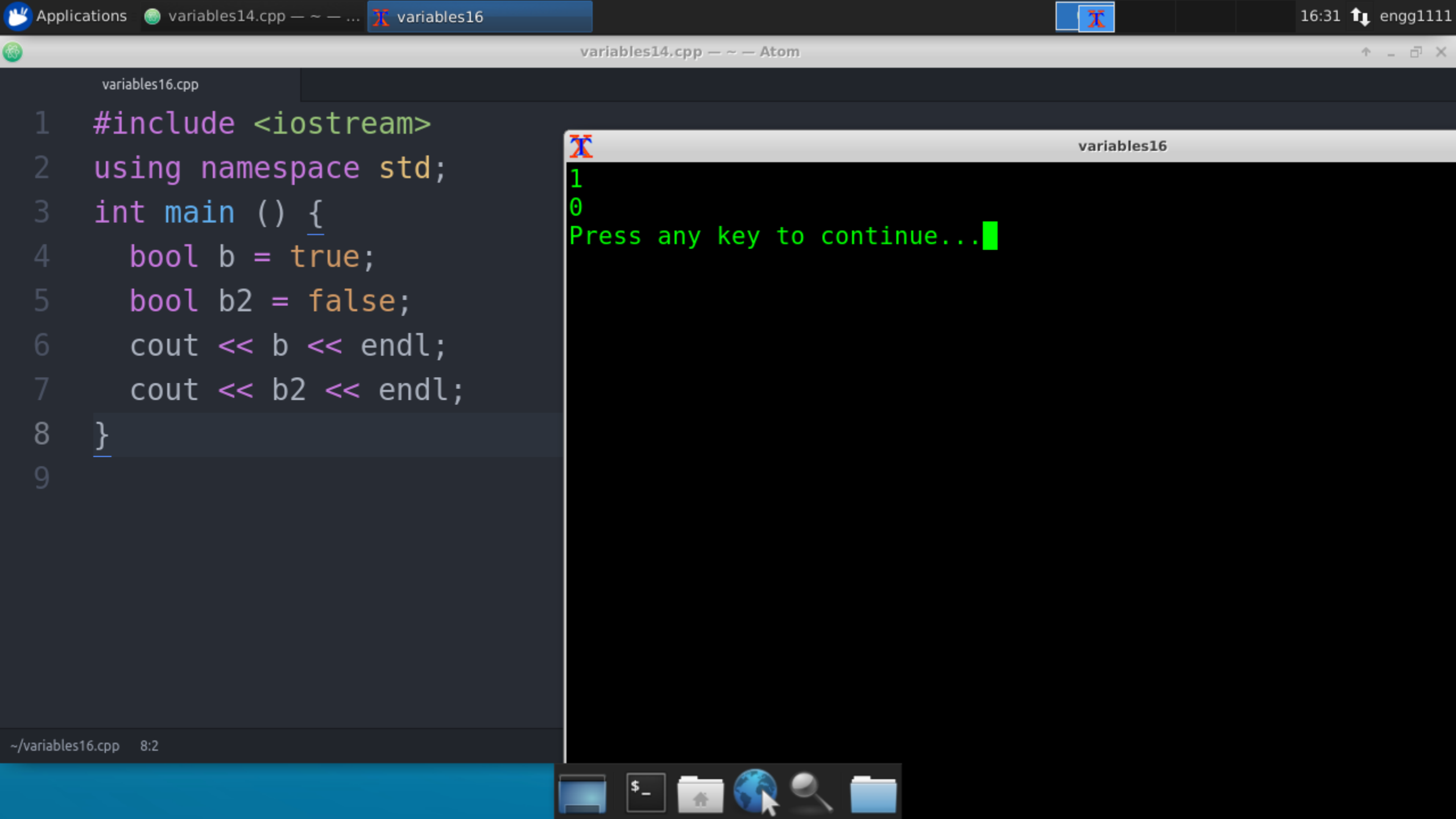
```
int age = 5, steps = age + 10;
char c = 'Y';
bool win = true;
double height = 120.5, length = 1.5e3;
```

- 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


Applications
variables13.cpp — ~/co...
variables13
16:22
engg1111
variables13.cpp — ~/code — Atom
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
variables13
5
15
Y
1
120.5
1500
Press any key to continue...


Applications
variables14.cpp — ~ — ...
variables14
16:28
engg1111
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
~/variables14.cpp 6:2
variables14
P
Press any key to continue...


Applications
variables14.cpp — ~ — ...
variables15
16:28
engg1111
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
~/variables15.cpp 4:14
variables15
P
Press any key to continue...

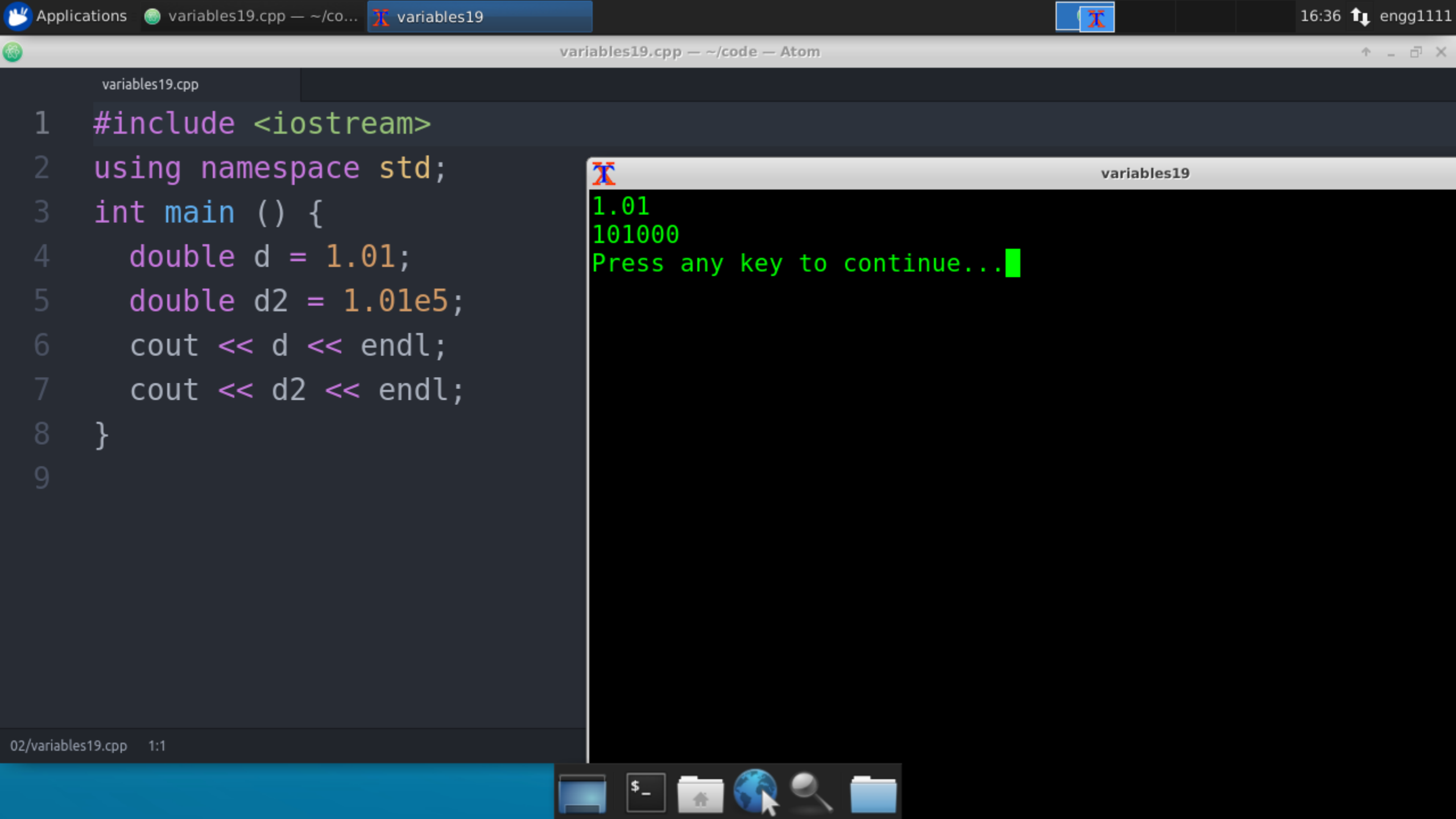

Applications
variables14.cpp — ~ — ...
variables16
16:31
engg1111
variables14.cpp — ~ — Atom
variables16.cpp
1 #include <iostream>
2 using namespace std;
3 int main () {
4 bool b = true;
5 bool b2 = false;
6 cout << b << endl;
7 cout << b2 << endl;
8 }
9
variables16
1
0
Press any key to continue...
~/variables16.cpp 8:2

variables17.cpp

```cpp
#include <iostream>
using namespace std;
int main () {
  bool b = 0;
  bool b2 = 1;
  bool b3 = -2;
  cout << b << endl;
  cout << b2 << endl;
  cout << b3 << endl;
}

```

variables18.cpp

```cpp
#include <iostream>
using namespace std;
int main () {
  bool b = true;
  bool b2 = false;
  bool b3 = true;
  b = b2;
  b2 = b3;
  b3 = false;
  cout << b << endl;
  cout << b2 << endl;
  cout << b3 << endl;
}

```


Applications
variables19.cpp — ~/co...
variables19
16:36
engg1111
variables19.cpp — ~/code — Atom
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
variables19
1.01
101000
Press any key to continue...
02/variables19.cpp 1:1

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

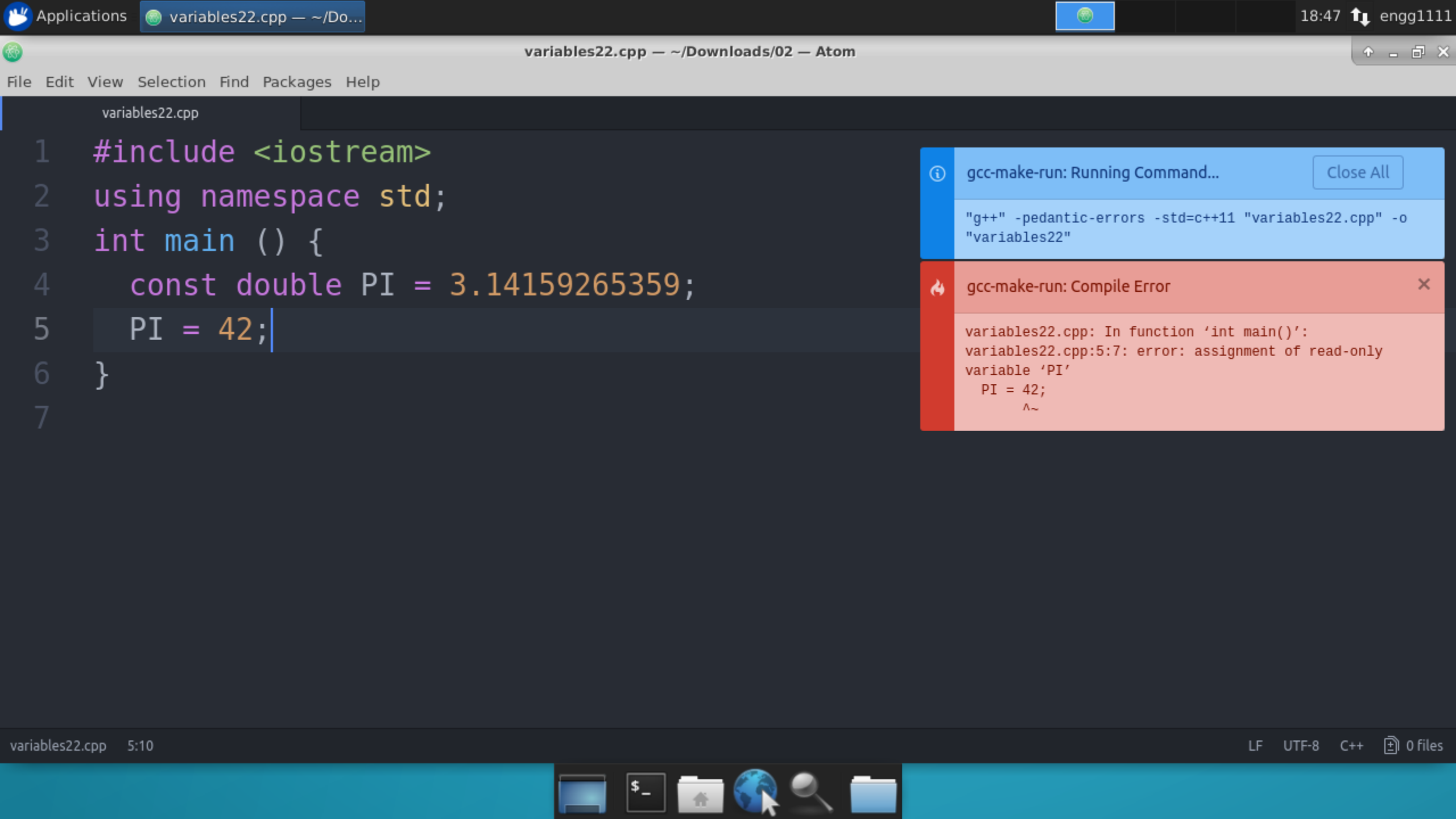
```
1  //Declaration
2  #include <iostream>
3  using namespace std;
4  int main () {
5    int a, b, c;
6    cout << a << " " << b << " " << c << endl;
7  }
8
```

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

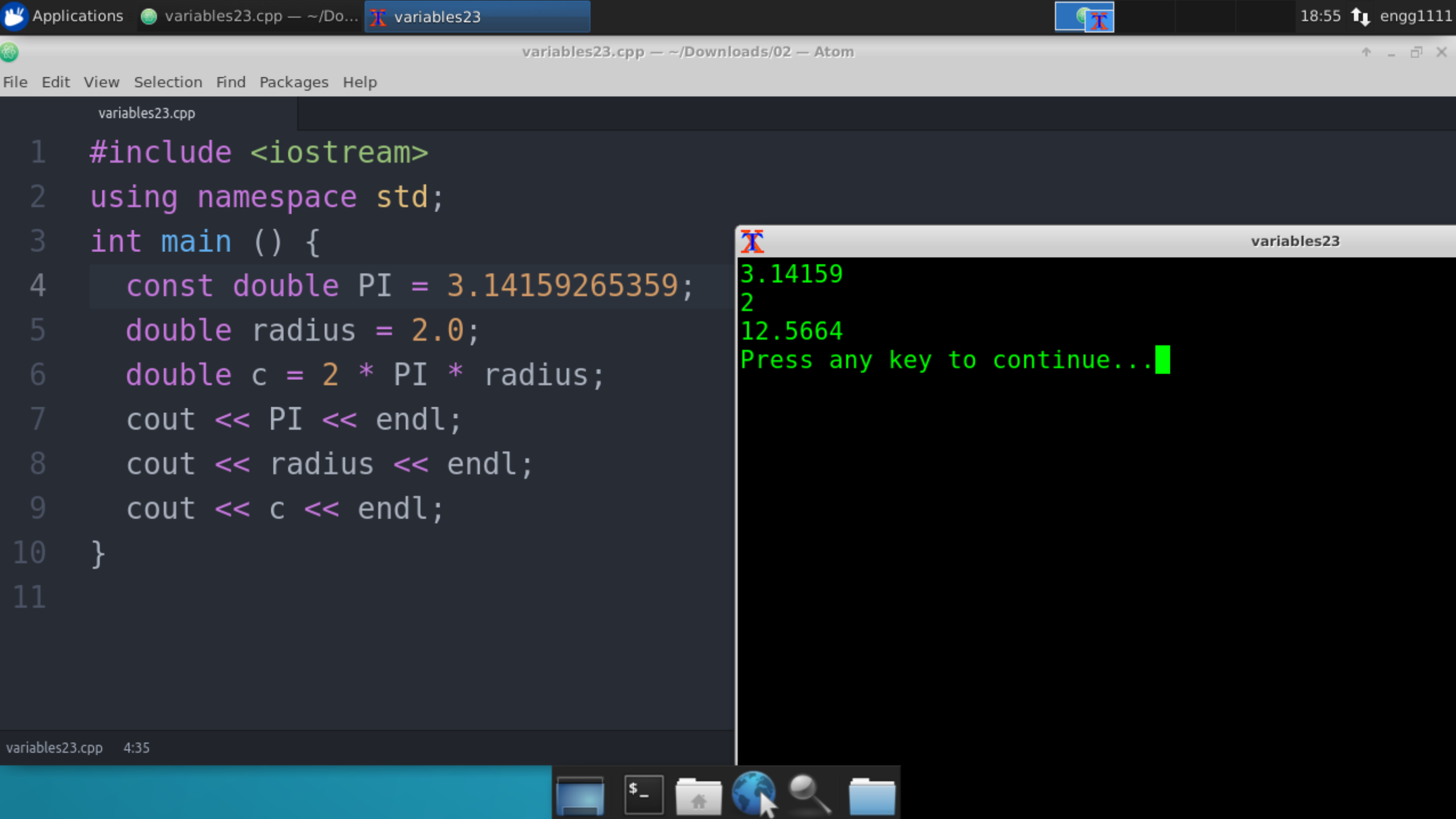

Applications
variables22.cpp — ~/Do...
18:47
engg1111
variables22.cpp — ~/Downloads/02 — Atom
File Edit View Selection Find Packages Help
variables22.cpp
1 #include <iostream>
2 using namespace std;
3 int main () {
4 const double PI = 3.14159265359;
5 PI = 42;
6 }
7
gcc-make-run: Running Command...
Close All
"g++" -pedantic-errors -std=c++11 "variables22.cpp" -o "variables22"
gcc-make-run: Compile Error
variables22.cpp: In function ‘int main()’:
variables22.cpp:5:7: error: assignment of read-only variable ‘PI’
PI = 42;
^~
variables22.cpp 5:10
LF UTF-8 C++ 0 files

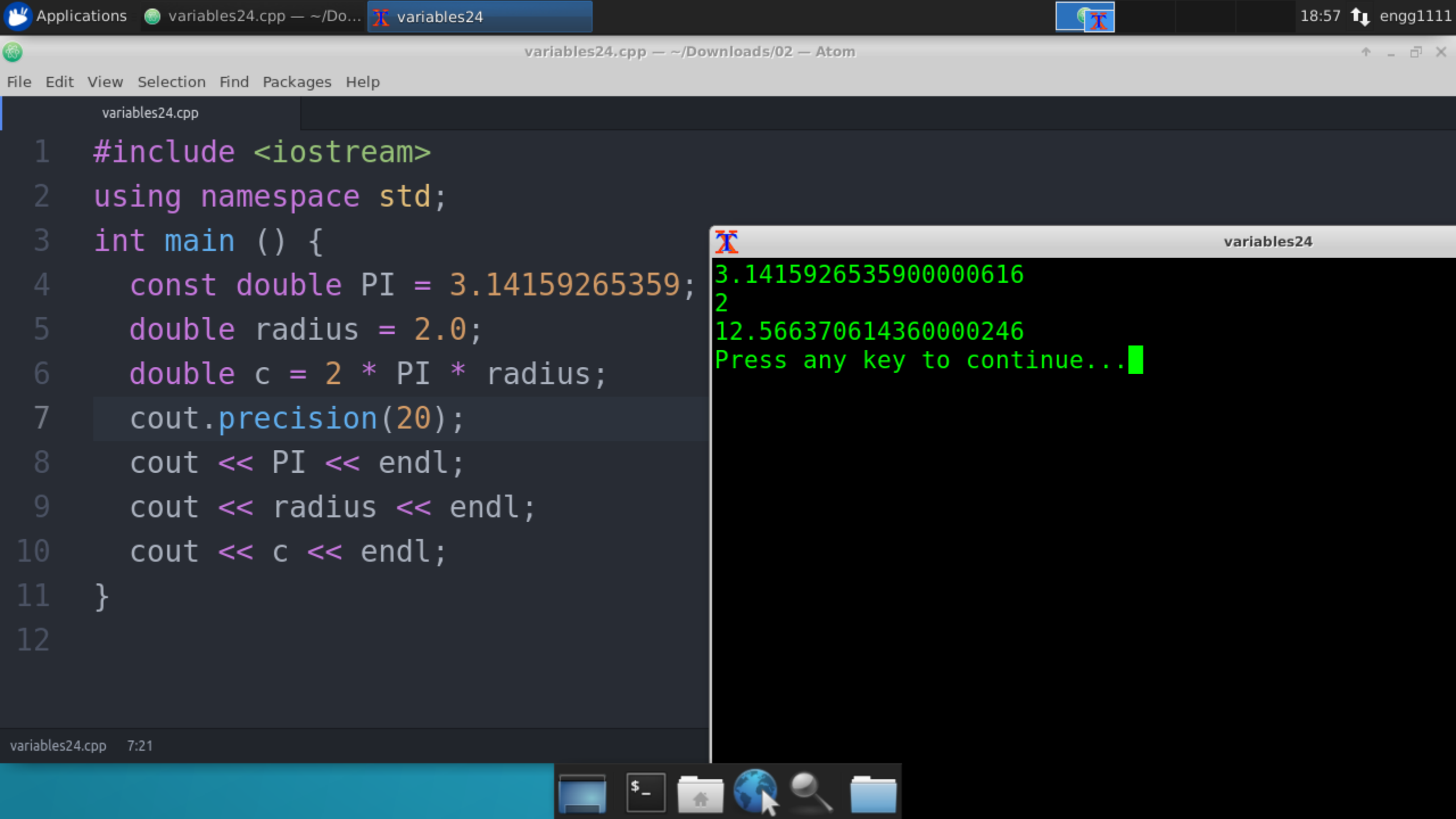
Expressions

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
```


Applications
variables23.cpp — ~/Do...
variables23
18:55
engg1111
variables23.cpp — ~/Downloads/02 — Atom
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
variables23.cpp 4:35
variables23
3.14159
2
12.5664
Press any key to continue...


Applications
variables24.cpp — ~/Do...
variables24
18:57
engg1111
variables24.cpp — ~/Downloads/02 — Atom
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
variables24.cpp 7:21
variables24
3.1415926535900000616
2
12.566370614360000246
Press any key to continue...

Operators

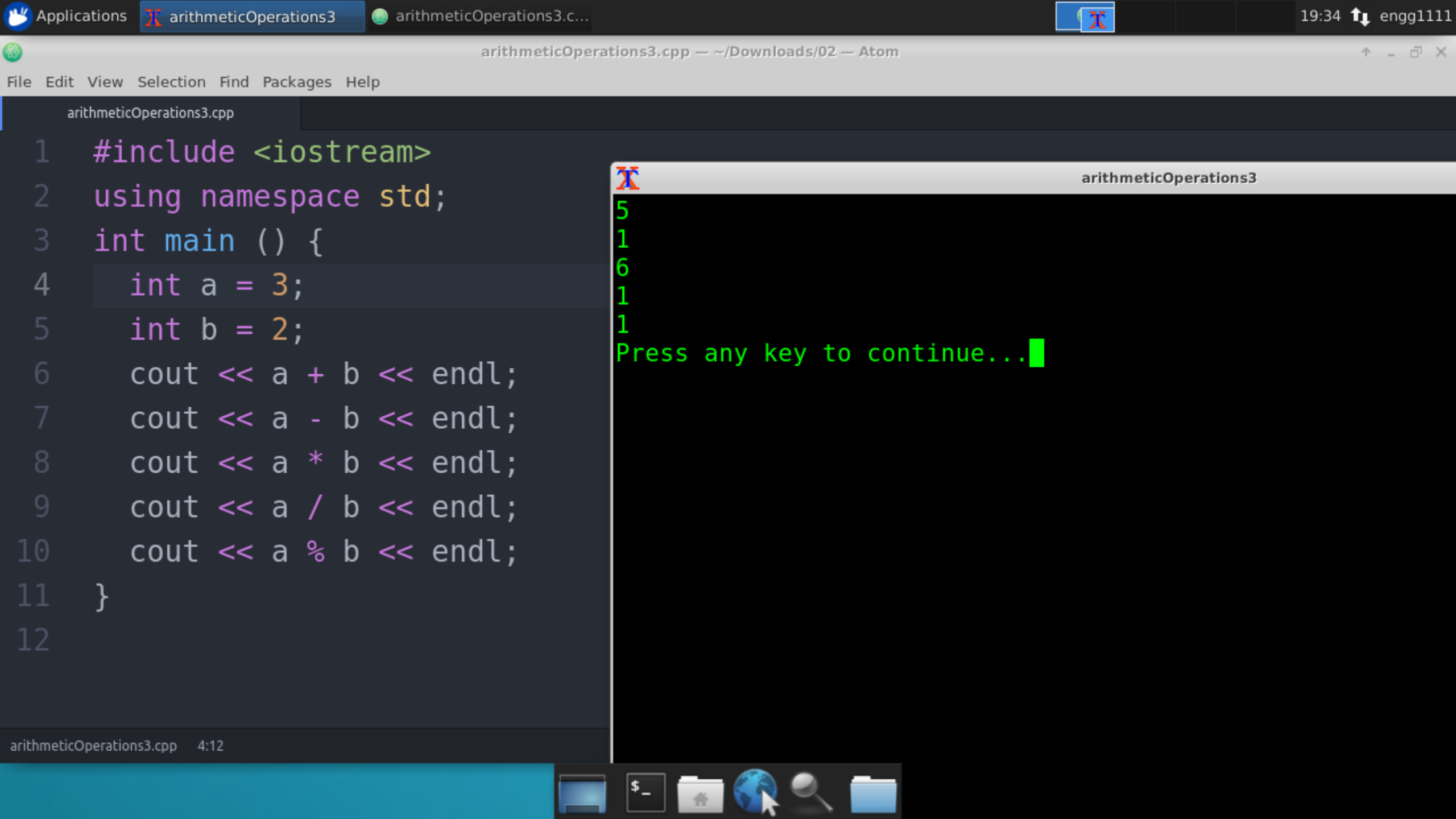
- Arithmetic Operators
- Relational Operators
- Logical Operators
- Increment and Decrement Operators
- Assignment Operators

Arithmetic Operators

Arithmetic Operators

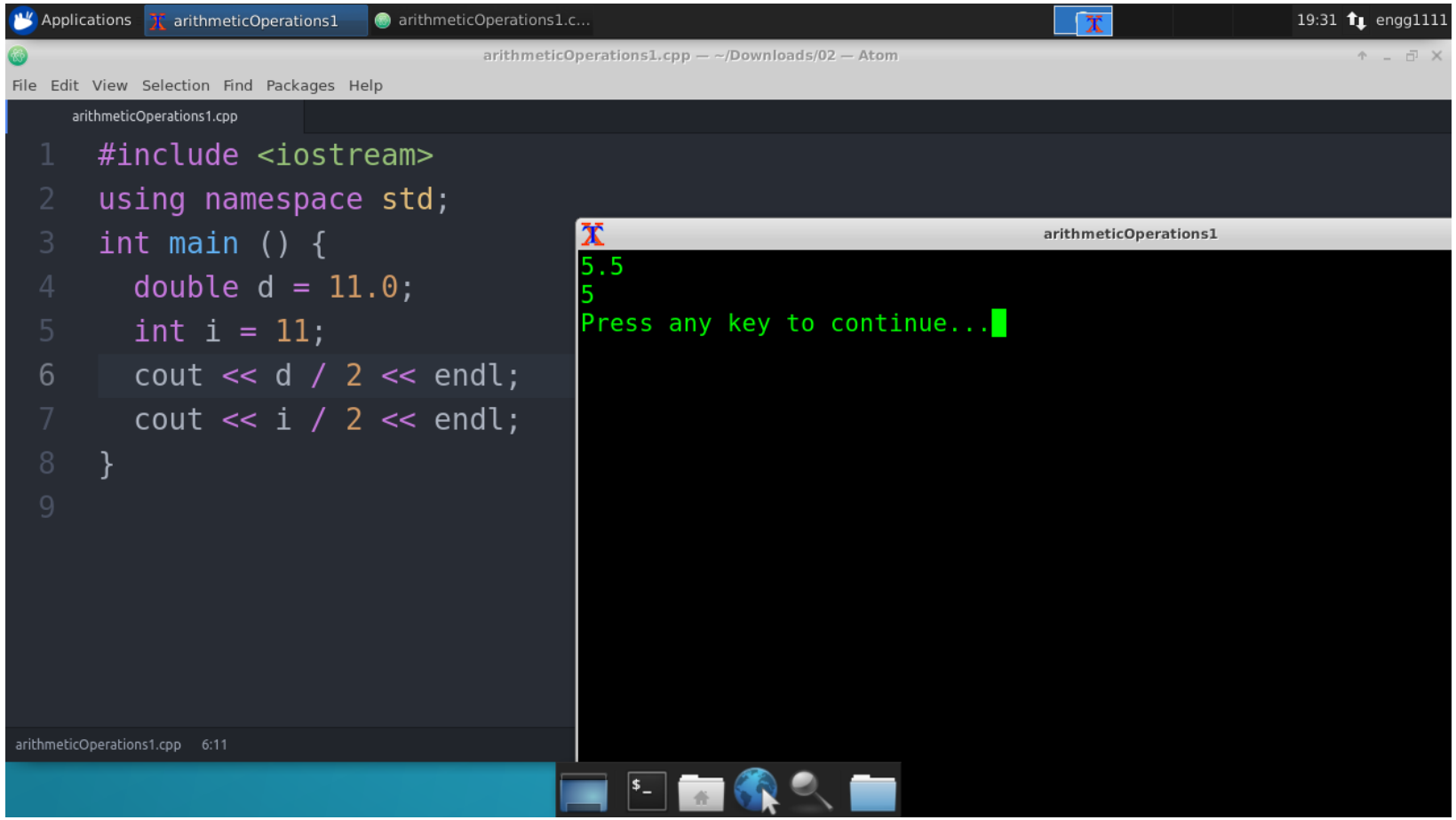
Arithmetic Operators	Sign in the expression
Addition	+
Subtraction	-
Multiplication	*
Division	/
Modulus	%

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


Applications
arithmeticOperations3
arithmeticOperations3.c...
19:34
engg1111
arithmeticOperations3.cpp — ~/Downloads/02 — Atom
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
arithmeticOperations3.cpp 4:12
arithmeticOperations3
5
1
6
1
1
Press any key to continue...

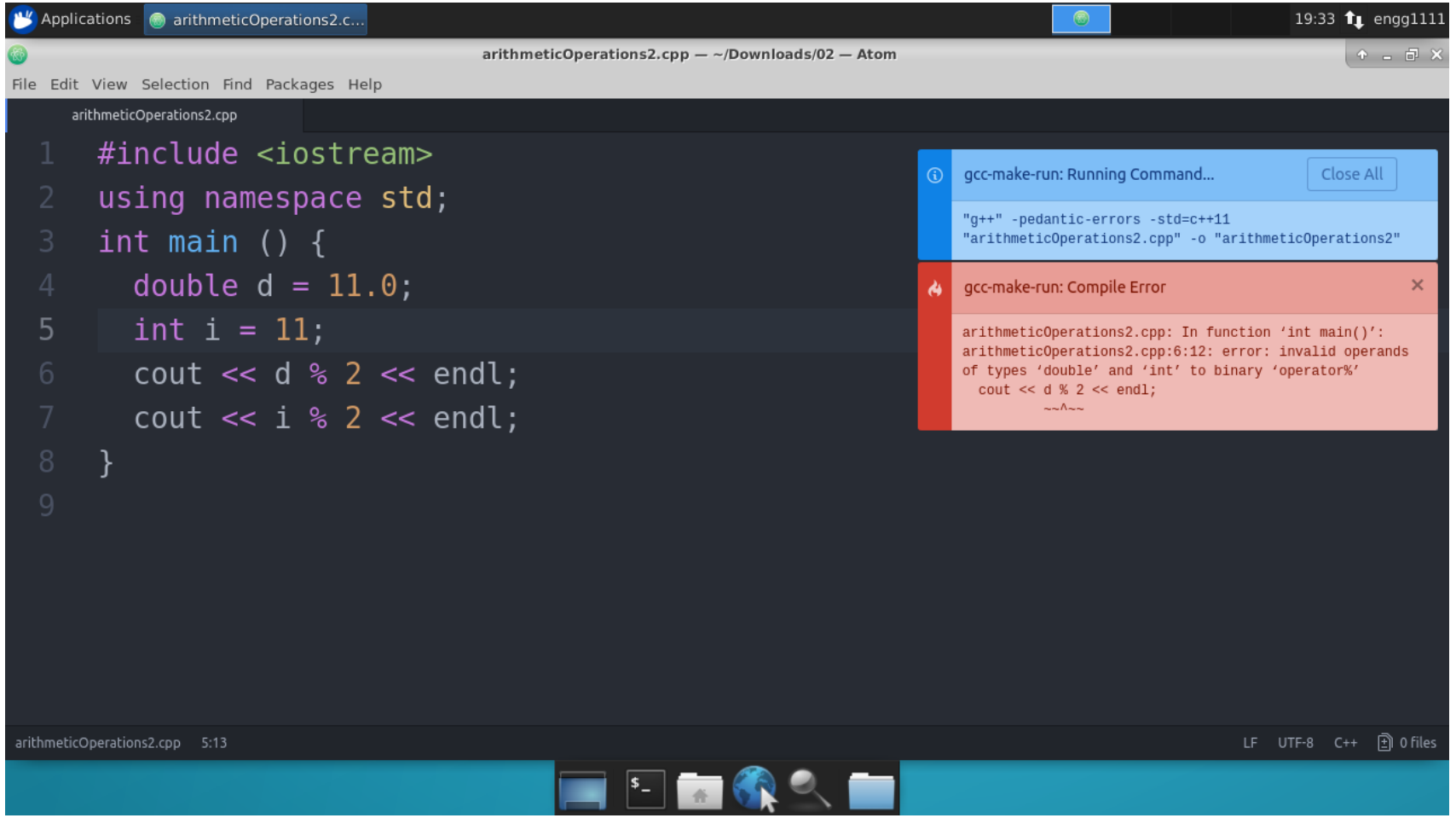
Arithmetic Operators

- Note that integer division truncates any fractional part



Arithmetic Operators

- The % operator cannot be applied to double



arithmeticOperations4.cpp — ~/Downloads/02 — Atom

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

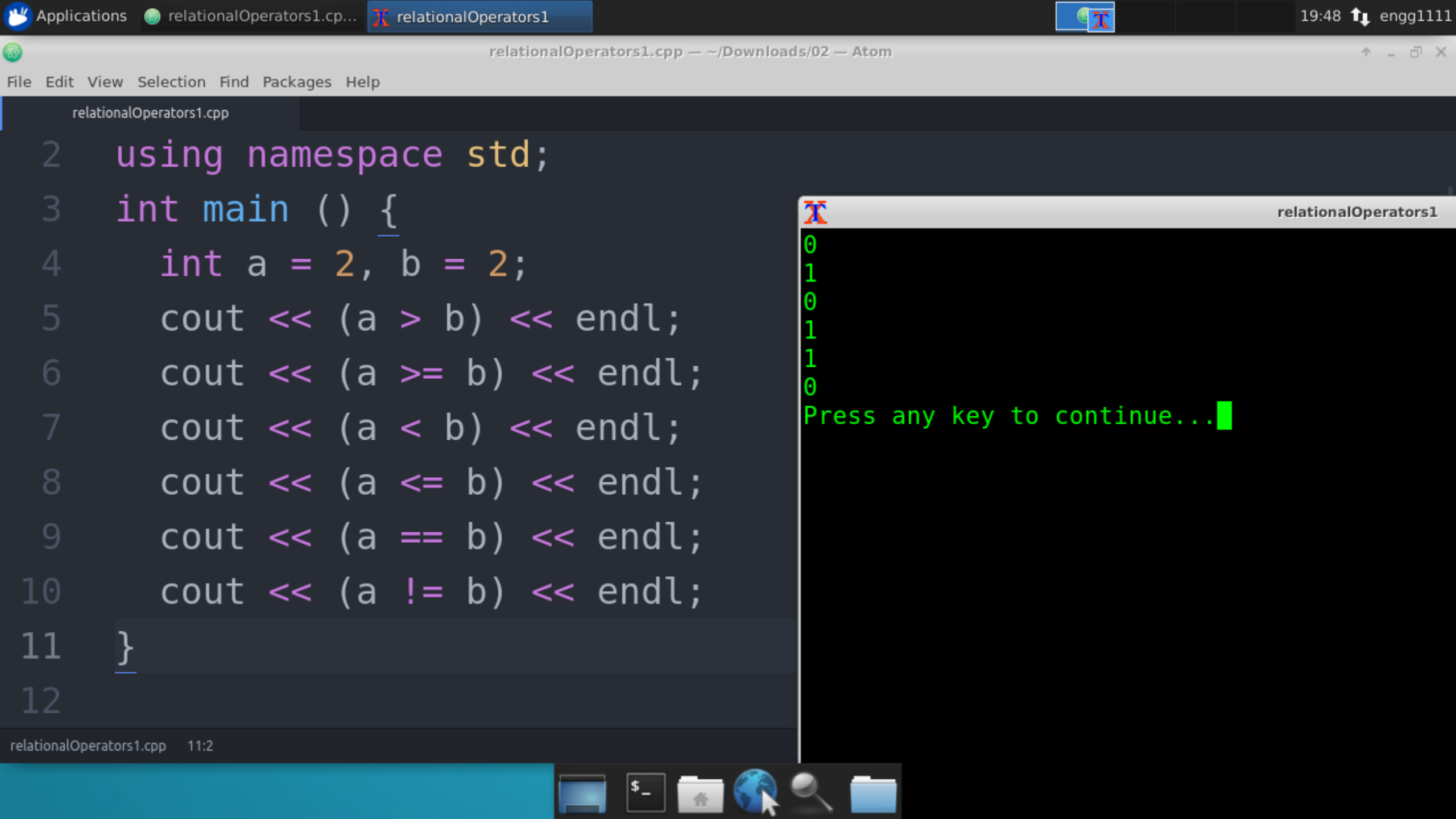
- The order of evaluation may be overridden by inserting parentheses () into the expressions
  - E.g., (1 + 2) * 3

Relational Operators

Relational Operators

Relational Operators	Sign in the expression
Greater than	>
Greater than or equal	>=
Smaller than	<
Smaller than or equal	<=
Equal	==
Not equal	!=

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


Applications
relationalOperators1.cp...
relationalOperators1
19:48
engg1111
relationalOperators1.cpp — ~/Downloads/02 — Atom
File Edit View Selection Find Packages Help
relationalOperators1.cpp
using namespace std;
int main () {
int a = 2, b = 2;
cout << (a > b) << endl;
cout << (a >= b) << endl;
cout << (a < b) << endl;
cout << (a <= b) << endl;
cout << (a == b) << endl;
cout << (a != b) << endl;
}
relationalOperators1.cpp 11:2
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

relationalOperators2.cpp — ~/Downloads/02 — Atom

File Edit View Selection Find Packages Help

relationalOperators2.cpp

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

relationalOperators3.cpp

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

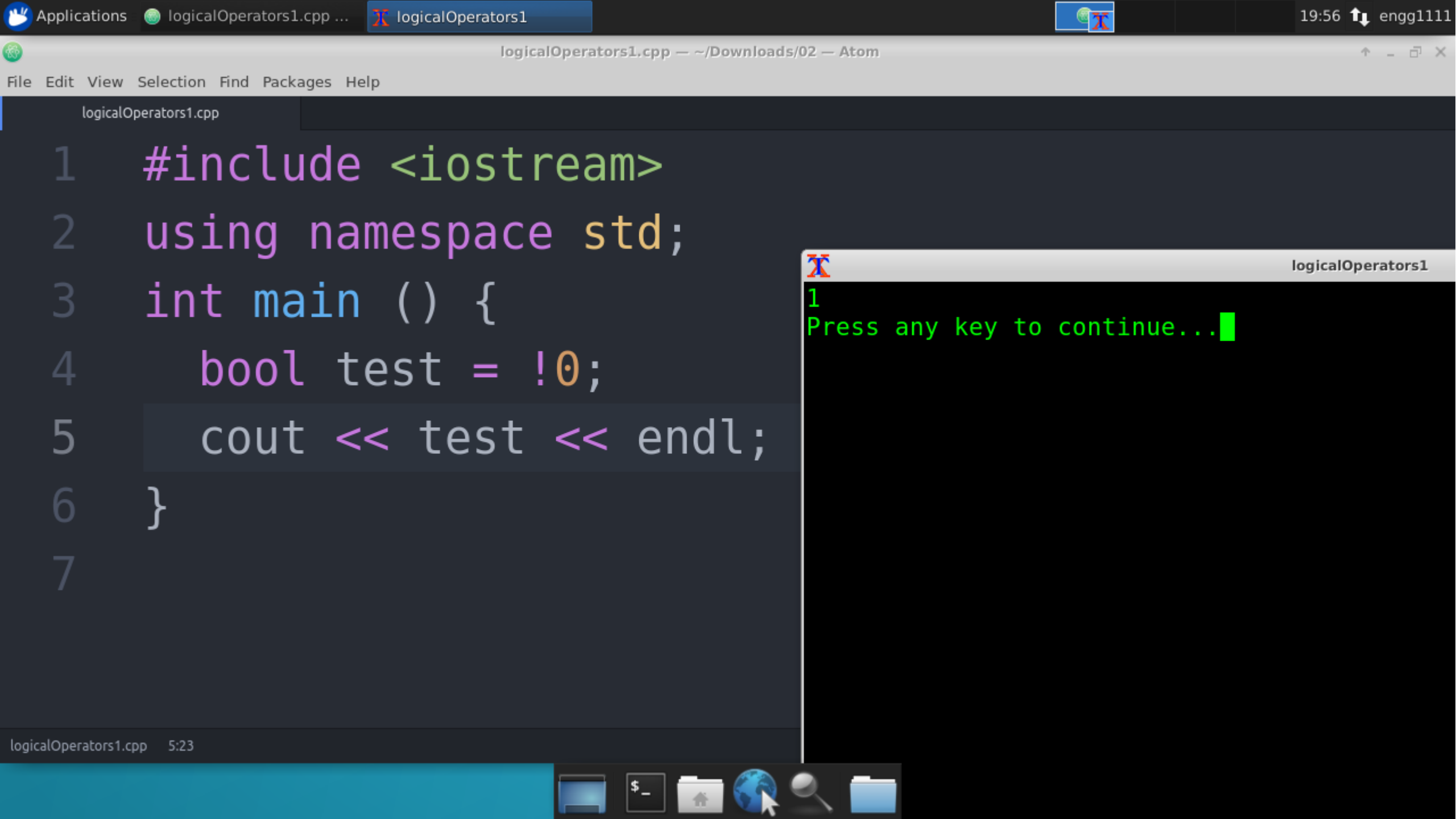
Logical Operators

Logical Operators

Logical Operators	Sign in the expression
And	&&
Or	||
Not	!

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


Applications
logicalOperators1.cpp ...
logicalOperators1
19:56
engg1111
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
logicalOperators1
1
Press any key to continue...

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

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

logicalOperators4.cpp

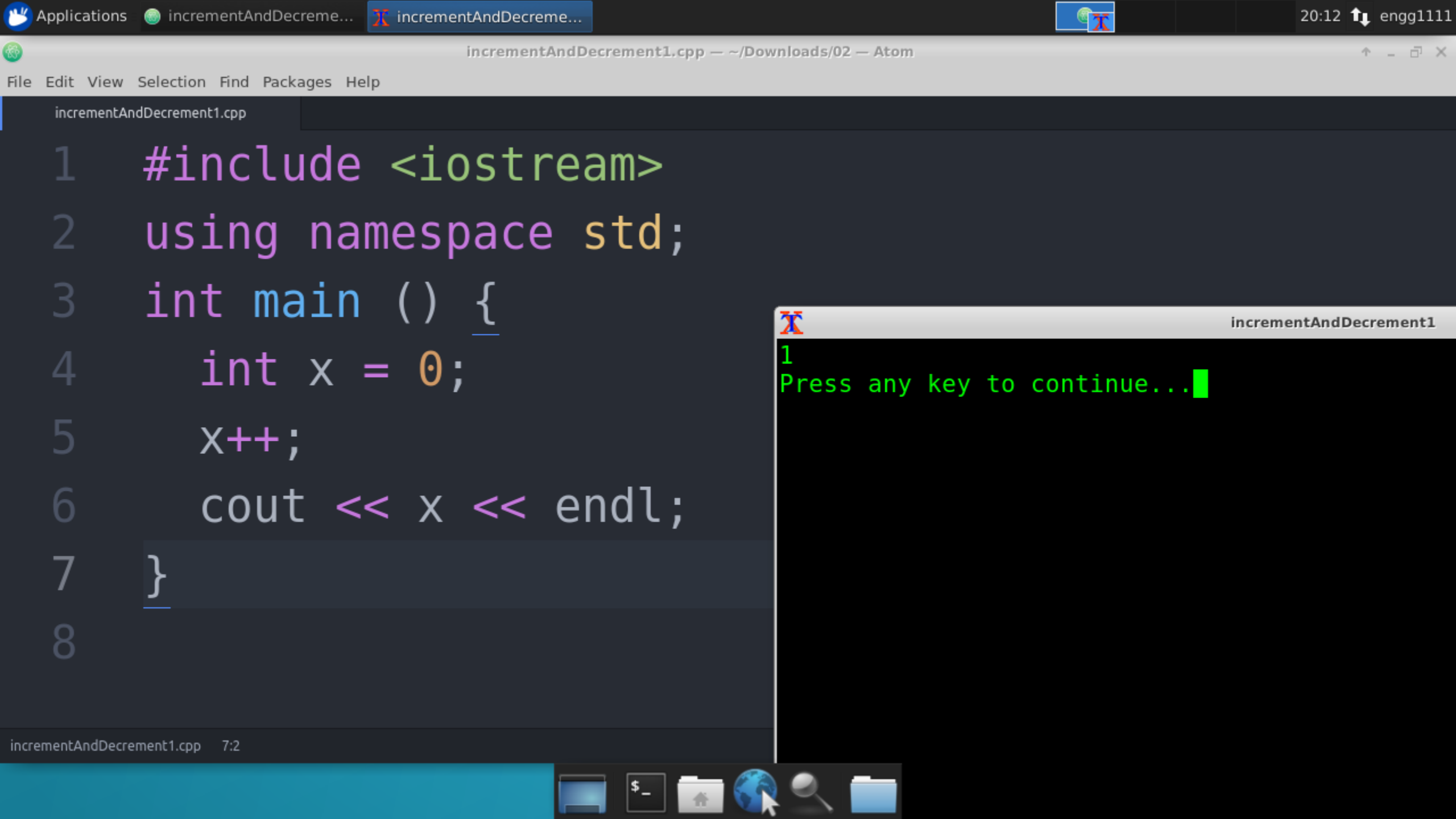
```cpp
#include <iostream>
using namespace std;
int main () {
  bool test = true || false && true;
  test = !test;
  cout << test << endl;
}

```

Increment and Decrement Operators

Increment and Decrement

Assignment Operators	Sign in the expression
Increment	++
Decrement	--

- 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


Applications
incrementAndDecreme...
incrementAndDecreme...
20:12
engg1111
incrementAndDecrement1.cpp — ~/Downloads/02 — Atom
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.cpp 7:2
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

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
```

incrementAndDecrement3.cpp

```cpp
#include <iostream>
using namespace std;
int main () {
   int x = 0;
   cout << x++ << endl;
   cout << x << endl;
}

```

incrementAndDecrement4.cpp

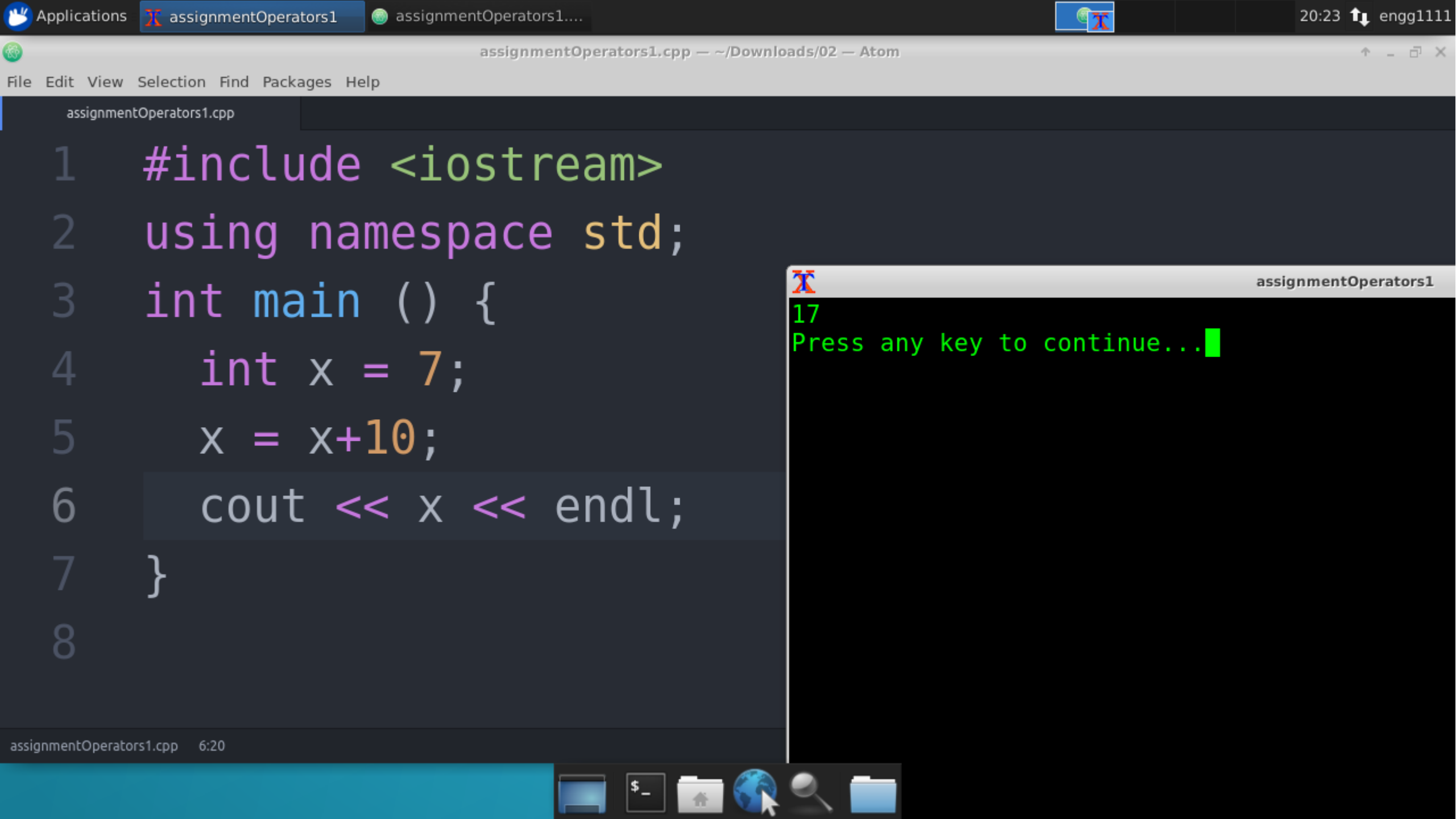
```
1  #include <iostream>
2  using namespace std;
3  int main () {
4     int x = 0;
5     cout << (false&&x++) << endl;
6     cout << x << endl;
7  }
8
```

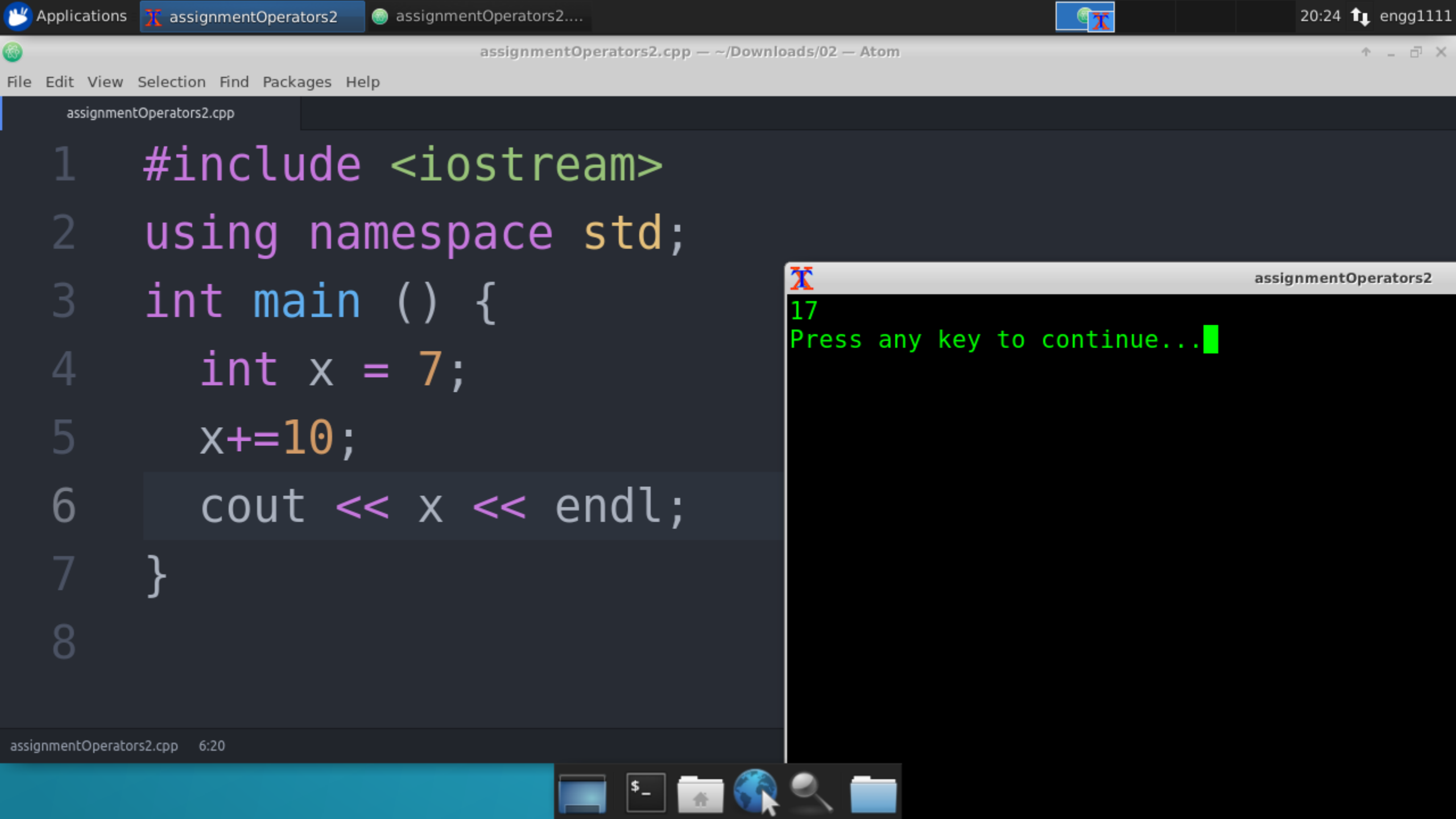
Assignment Operators

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


Applications
assignmentOperators1
assignmentOperators1....
20:23
engg1111
assignmentOperators1.cpp — ~/Downloads/02 — Atom
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
assignmentOperators1.cpp 6:20
assignmentOperators1
17
Press any key to continue...

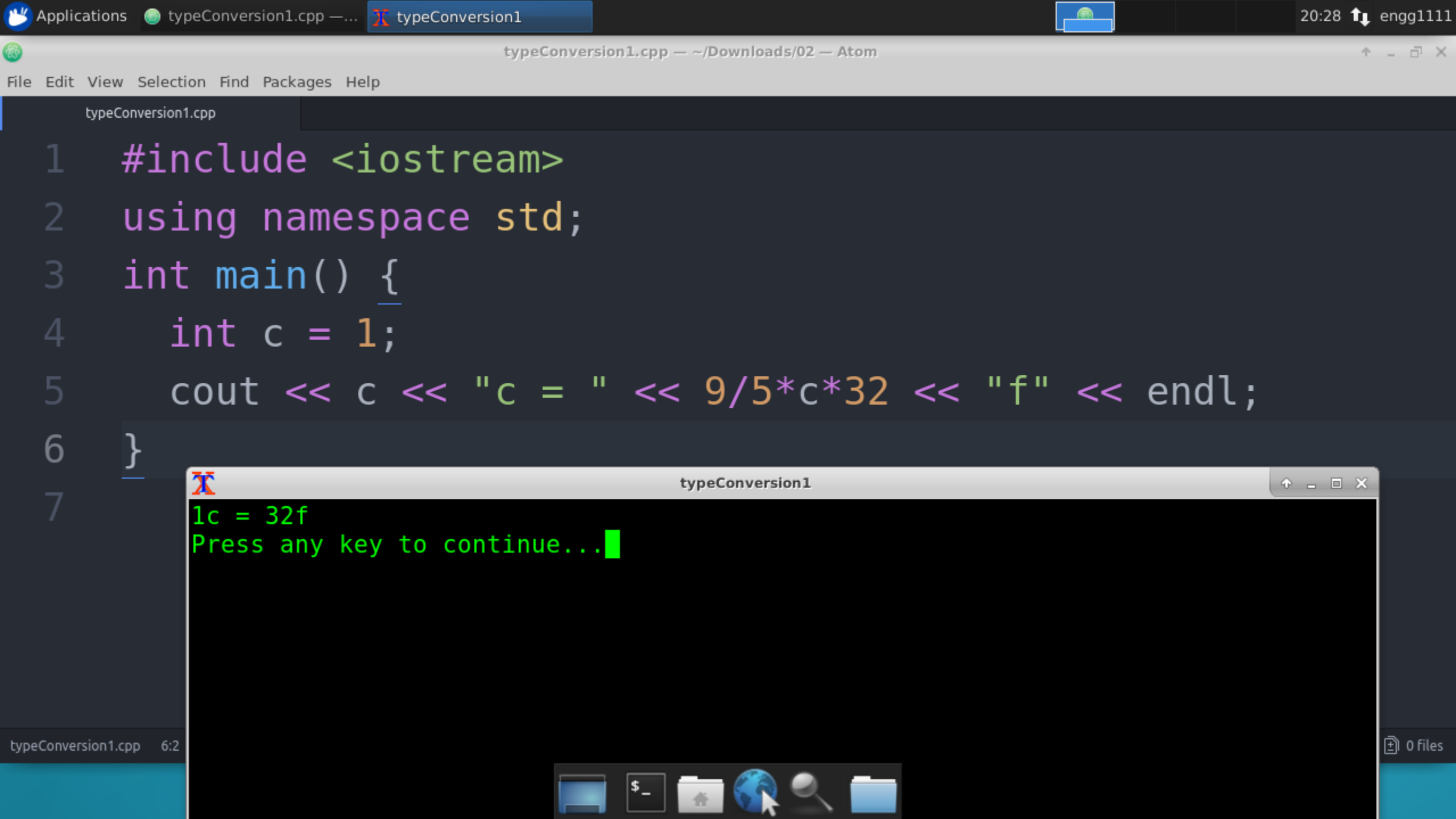

Applications
assignmentOperators2
assignmentOperators2....
20:24
engg1111
assignmentOperators2.cpp — ~/Downloads/02 — Atom
File Edit View Selection Find Packages Help
assignmentOperators2.cpp
1 #include <iostream>
2 using namespace std;
3 int main () {
4 int x = 7;
5 x+=10;
6 cout << x << endl;
7 }
8
assignmentOperators2.cpp 6:20
assignmentOperators2
17
Press any key to continue...

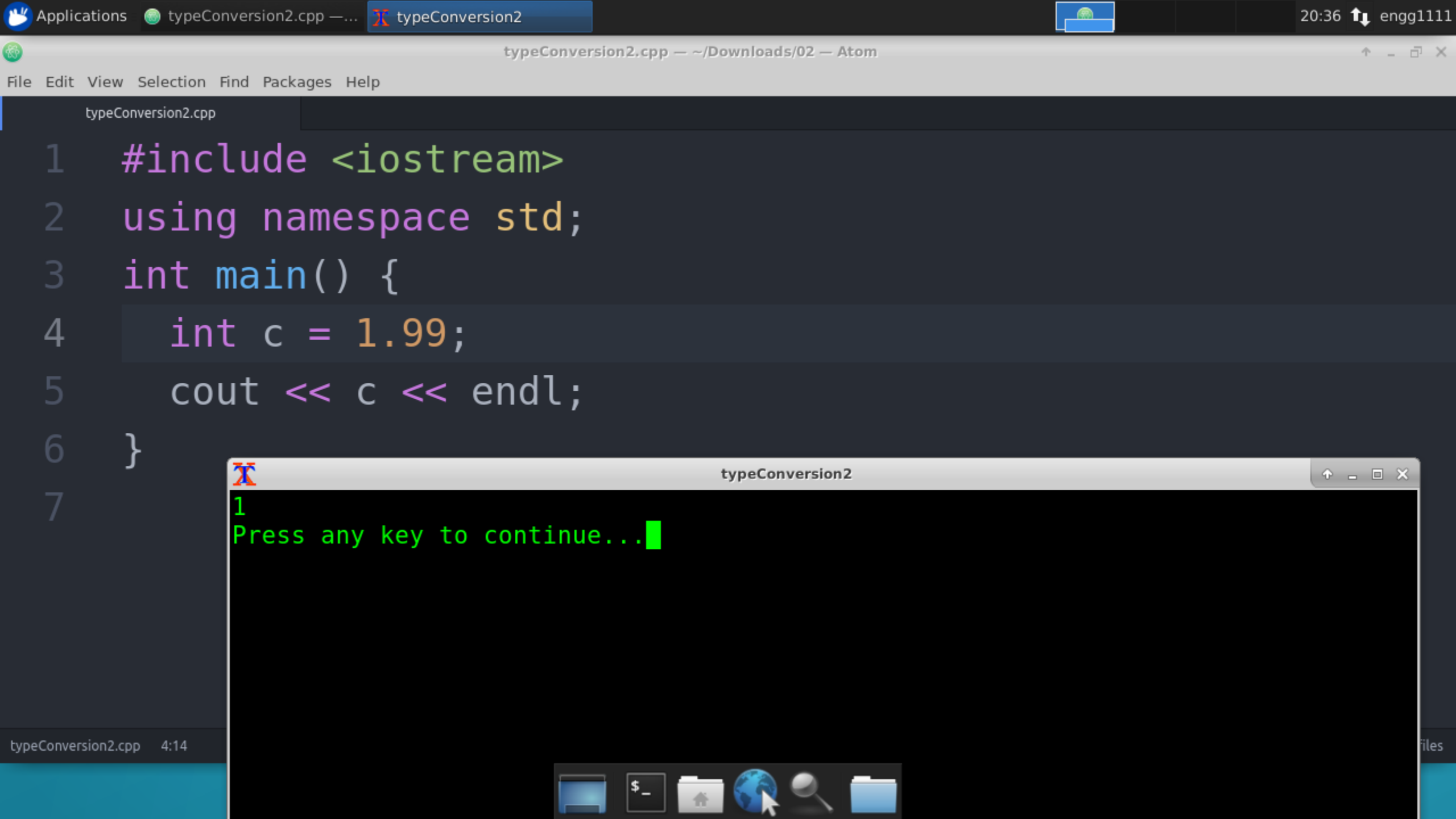
Type Conversion

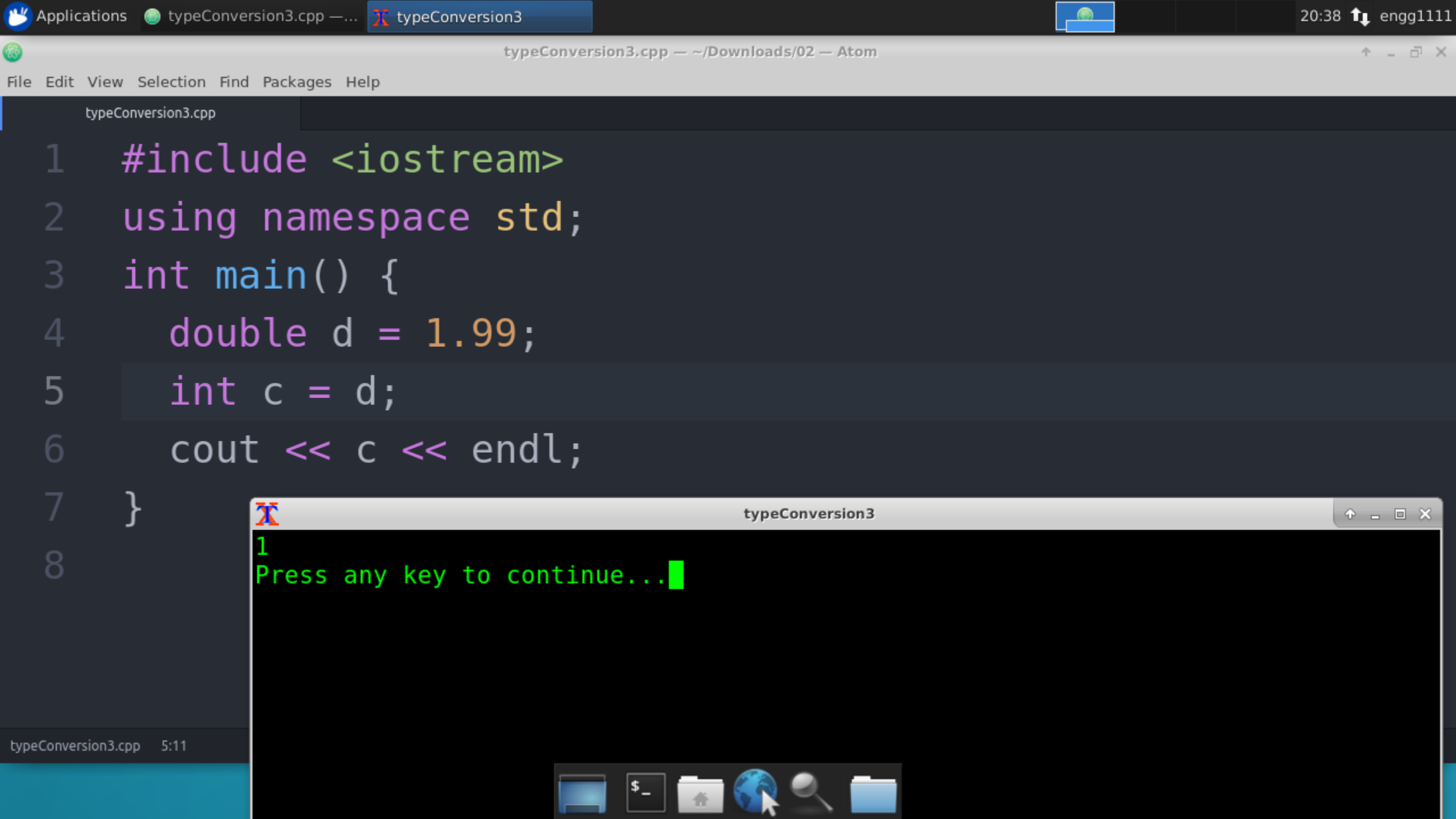
Task

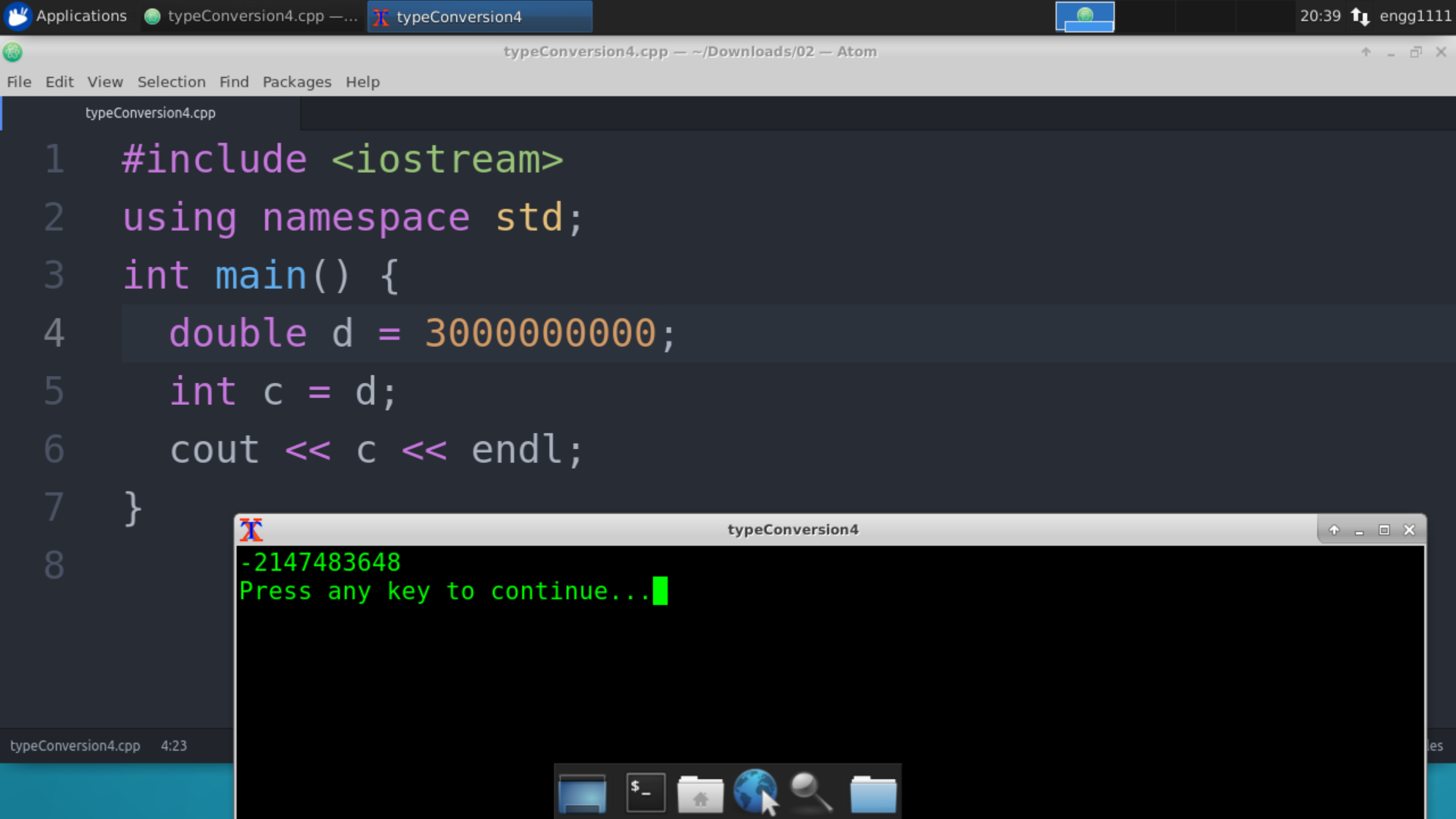
- Write a program to that converts degree Celsius to Fahrenheit

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


Applications
typeConversion1.cpp —...
typeConversion1
20:28
engg1111
typeConversion1.cpp — ~/Downloads/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
typeConversion1
1c = 32f
Press any key to continue...
typeConversion1.cpp 6:2
0 files

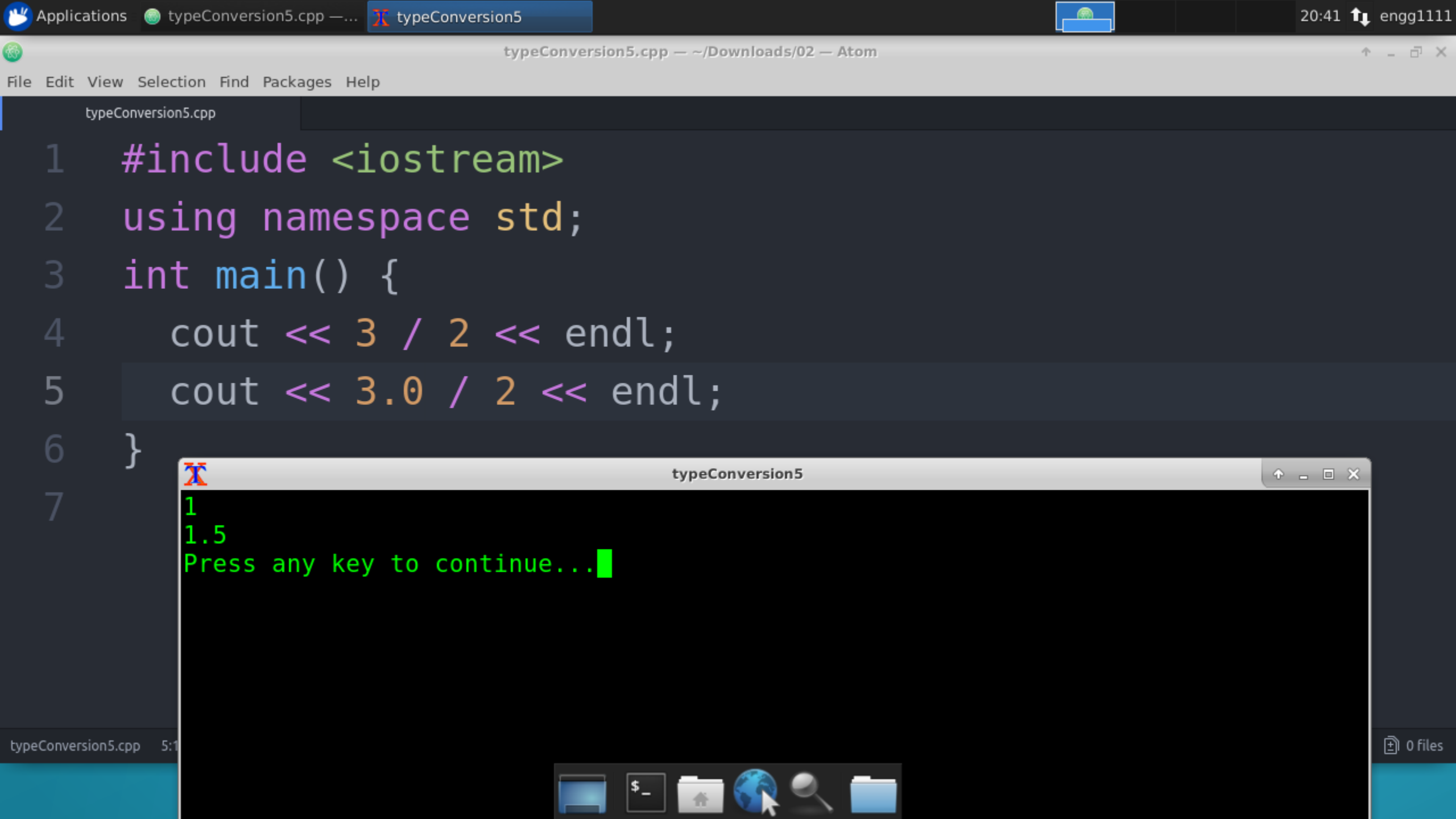

Applications
typeConversion2.cpp —...
typeConversion2
20:36
engg1111
typeConversion2.cpp — ~/Downloads/02 — Atom
File Edit View Selection Find Packages Help
typeConversion2.cpp
1 #include <iostream>
2 using namespace std;
3 int main() {
4 int c = 1.99;
5 cout << c << endl;
6 }
7
typeConversion2
1
Press any key to continue...
typeConversion2.cpp 4:14


Applications
typeConversion3.cpp —...
typeConversion3
20:38
engg1111
typeConversion3.cpp — ~/Downloads/02 — Atom
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 }
8
typeConversion3.cpp 5:11
typeConversion3
1
Press any key to continue...


Applications
typeConversion4.cpp —...
typeConversion4
20:39
engg1111
typeConversion4.cpp — ~/Downloads/02 — Atom
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 }
8
typeConversion4
-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
20:41
engg1111
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
typeConversion5
1
1.5
Press any key to continue...
typeConversion5.cpp 5:1
0 files

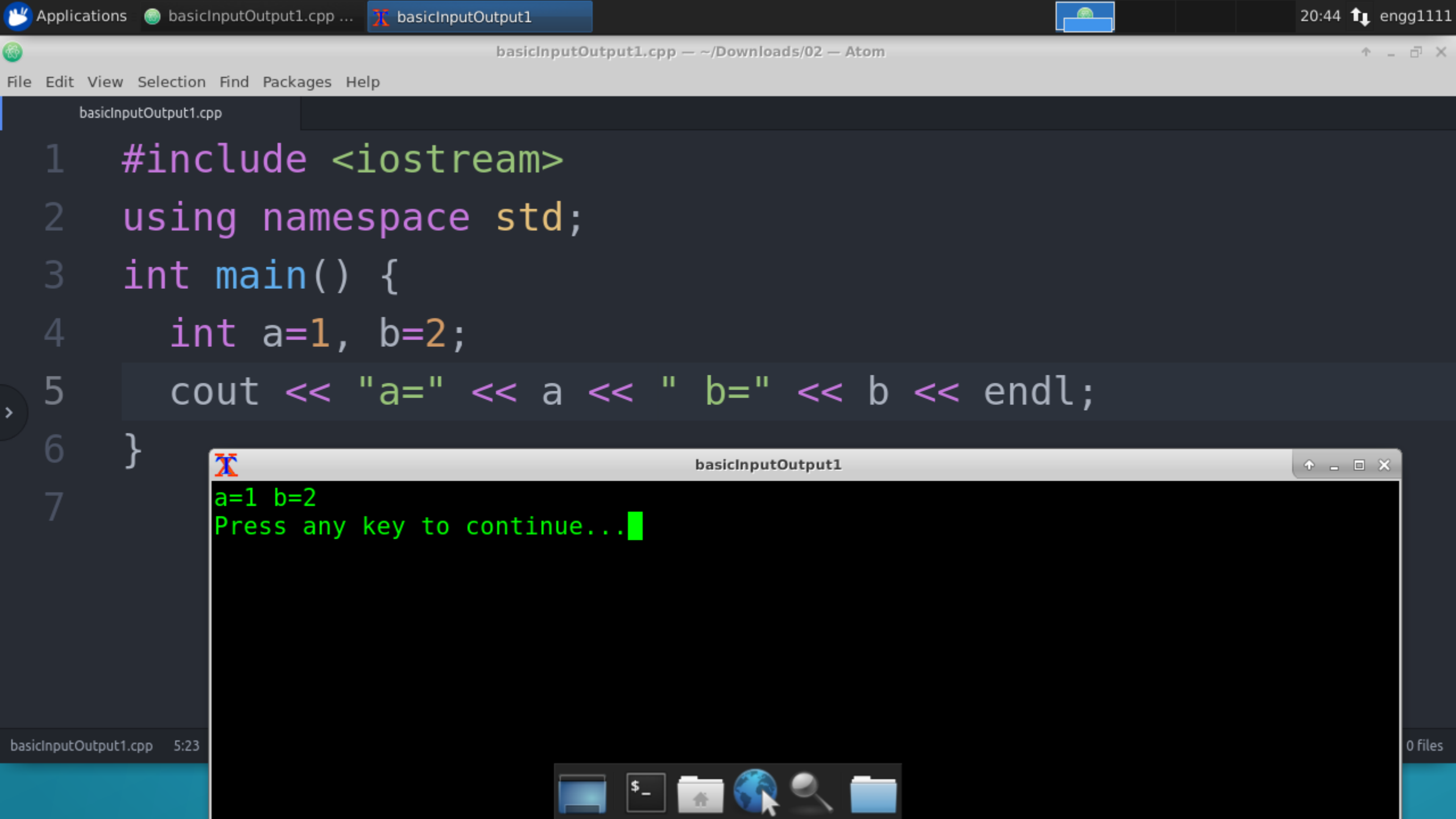
Basic Input / Output

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


Applications
basicInputOutput1.cpp ...
basicInputOutput1
20:44
engg1111
basicInputOutput1.cpp — ~/Downloads/02 — Atom
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
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

basicInputOutput2.cpp — ~/Downloads/02 — Atom

File Edit View Selection Find Packages Help

basicInputOutput2.cpp

```cpp
#include <iostream>
using namespace std;
int main() {
  int age, height, weight;
  cin >> age >> height >> weight;
  cout << "age=" << age << endl;
  cout << "height=" << height << endl;
  cout << "weight=" << weight << endl;
}

```

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