§4 Functions

ENGG1111

Computer Programming and Applications

Dirk Schnieders

Outline

- Top-down design approach
- Function
- Local and global variables
- Scope of variables

Top-down design approach

Top-Down Design Approach

- A good way to design a program is to break down the task to be accomplished into a few sub-tasks
- Each sub-task might be further decomposed into smaller sub-tasks, and this process is repeated until all sub-tasks are small enough that their implementations become manageable
- This approach is called top-down design (aka divide and conquer)

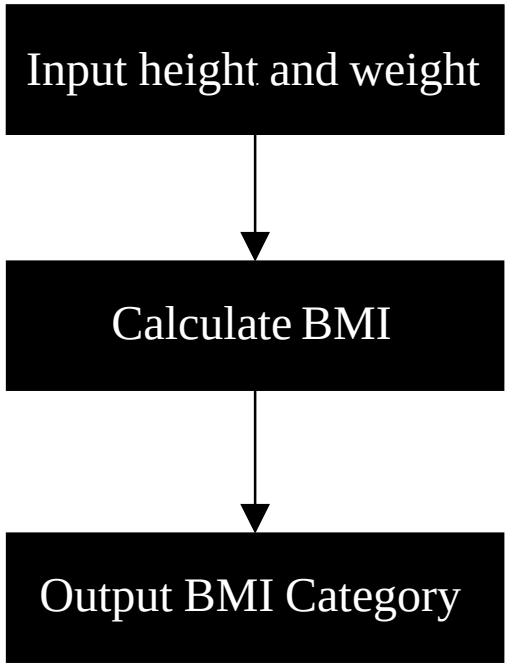
Example - BMI

- Write a program that reads in the weight (in kg) and height (in meter) of a user, and outputs the Body Mass Index (BMI) category
  - BMI = weight / height^2

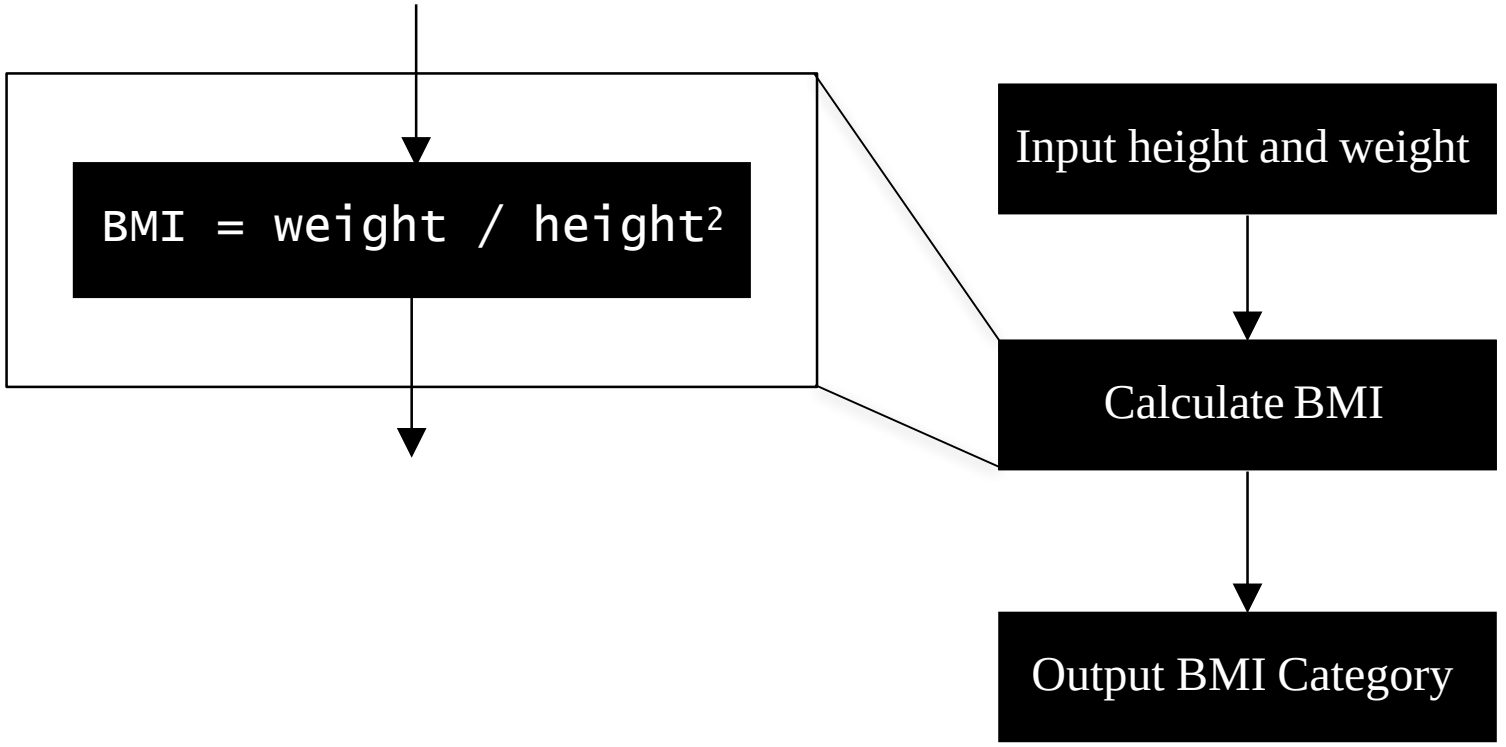
BMI	Category
<18.5	Underweight
[18.5, 22.9]	Normal
>22.9	Overweight

Example - BMI

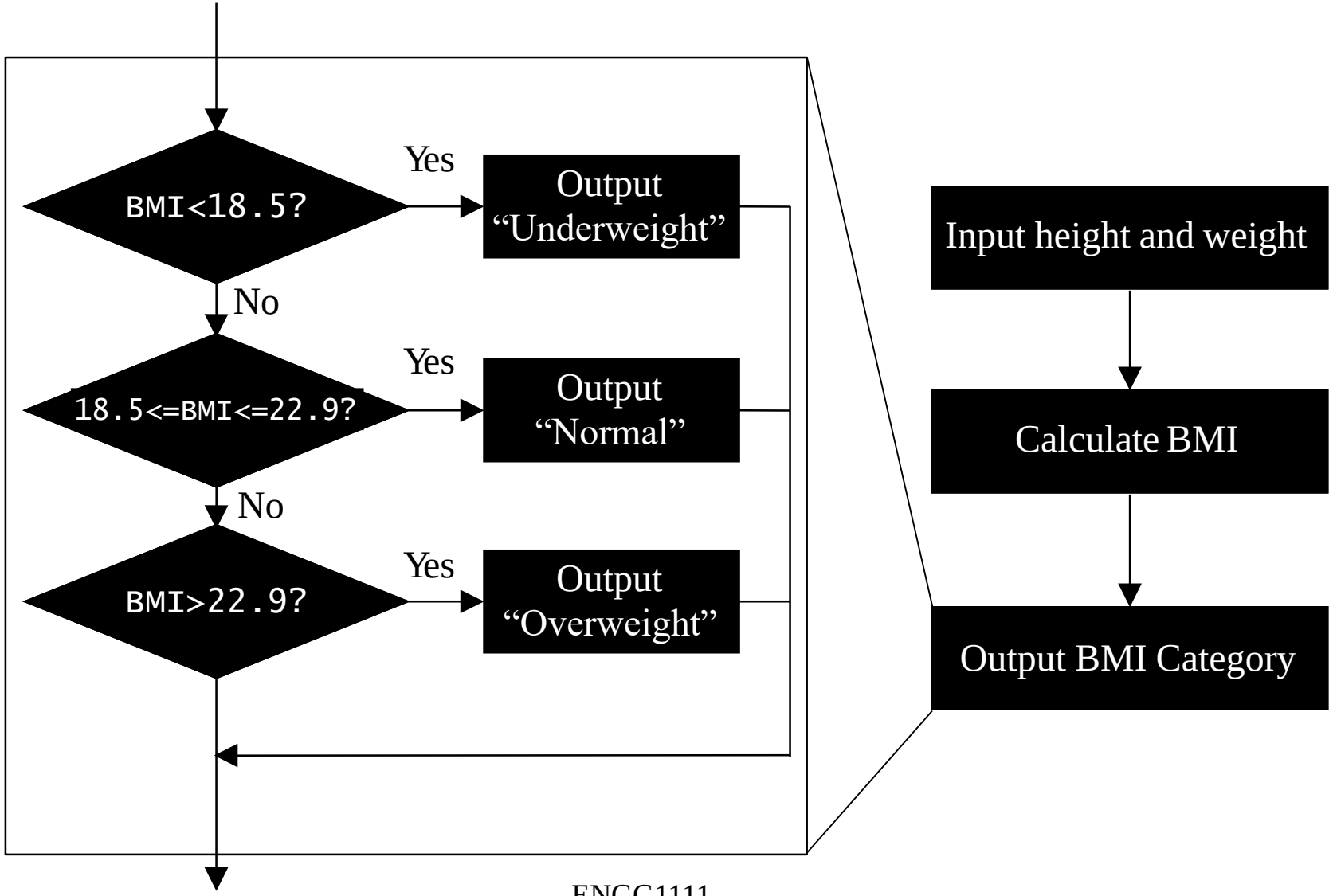
- We tackle the problem by dividing our program into three sub-tasks
- We will treat each part as a smaller problem and tackle (conquer) the smaller problems one by one



Example - BMI


BMI = weight / height²
Input height and weight
Calculate BMI
Output BMI Category

Example - BMI


BMI<18.5?
Yes
Output "Underweight"
No
18.5<=BMI<=22.9?
Yes
Output "Normal"
No
BMI>22.9?
Yes
Output "Overweight"
Input height and weight
Calculate BMI
Output BMI Category

Function

Functions

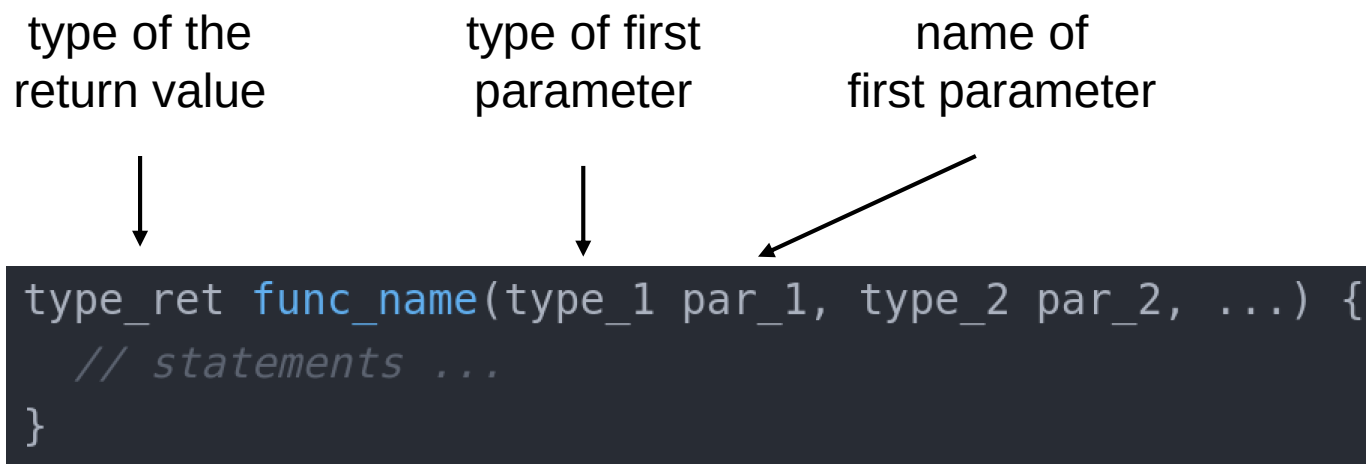
- Preserving the top-down design structure in a program will make it easier to understand, write and change the program
- In C++ sub-tasks can be implemented as functions
  - A function is a group of statements that is executed when it is called from some point in program
- A program is composed of a collection of functions
- When a program is put into execution, it always starts at the main function, which may in turn call other functions

Function Definition

- Describes how a function computes the value it returns
- Consists of a function header followed by a function body
  - The function header specifies
    - the type of the return value
    - the function name (identifier), and
    - the list of parameters (with types and identifiers)
  - The function body consists of variable statements enclosed within a pair of braces { }

header →
body →

```
type_ret func_name(type_1 par_1, type_2 par_2, ...) {
  // statements ...
}
```


type of the
return value
type of first
parameter
name of
first parameter
type_ret func_name(type_1 par_1, type_2 par_2, ...) {
// statements ...
}

return statement

- The value returned by the function is determined when the function executes a return statement
- A return statement consist of the keyword return followed by an expression
- When a return statement is executed the function call ends
- Note that a function may contain more than one return statement

functions02.cpp — ~/code/04 — Atom

File Edit View Selection Find Packages Help

functions02.cpp

```
1  #include <iostream>
2  using namespace std;
3  double celsiusToFahrenheit(double celsius) {
4      return 9.0/5*celsius+32;
5  }
6  int main() {
7  }
8
```

return statement

- Note that the main function returns the value 0 if a return statement is missing

functions03.cpp — ~/code/04 — Atom

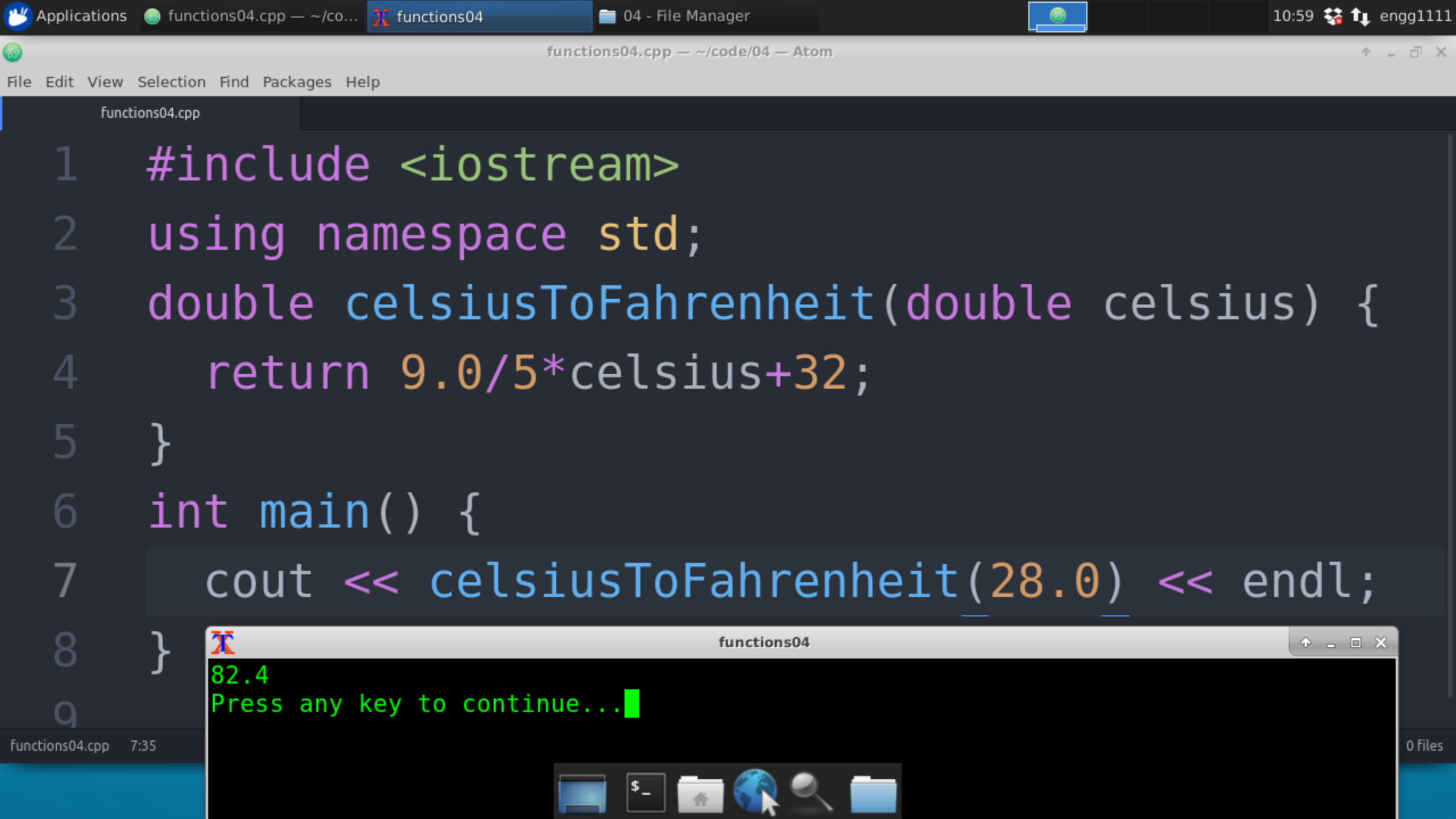
File Edit View Selection Find Packages Help

functions03.cpp

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      cout << "Hello World!" << endl;
5      return 0;
6  }
7
```

Function Call

- A function call (i.e., the process of calling a function) is made using the function name with the necessary parameters
- A function call is itself an expression, and can be put in any places where an expression is expected


Applications
functions04.cpp — ~/co...
functions04
04 - File Manager
10:59
engg1111
functions04.cpp — ~/code/04 — Atom
File Edit View Selection Find Packages Help
functions04.cpp
1 #include <iostream>
2 using namespace std;
3 double celsiusToFahrenheit(double celsius) {
4 return 9.0/5*celsius+32;
5 }
6 int main() {
7 cout << celsiusToFahrenheit(28.0) << endl;
8 }
9
functions04
82.4
Press any key to continue...
functions04.cpp 7:35
0 files

Parameters and Arguments

- Parameters are variables that are part of the function definition
- The expressions used to pass values to a function during a function call are referred to as arguments
- When a function is called the computer substitutes the first argument with the first parameter, the second argument with the second parameter, and so forth

functions05.cpp — ~/code/04 — Atom

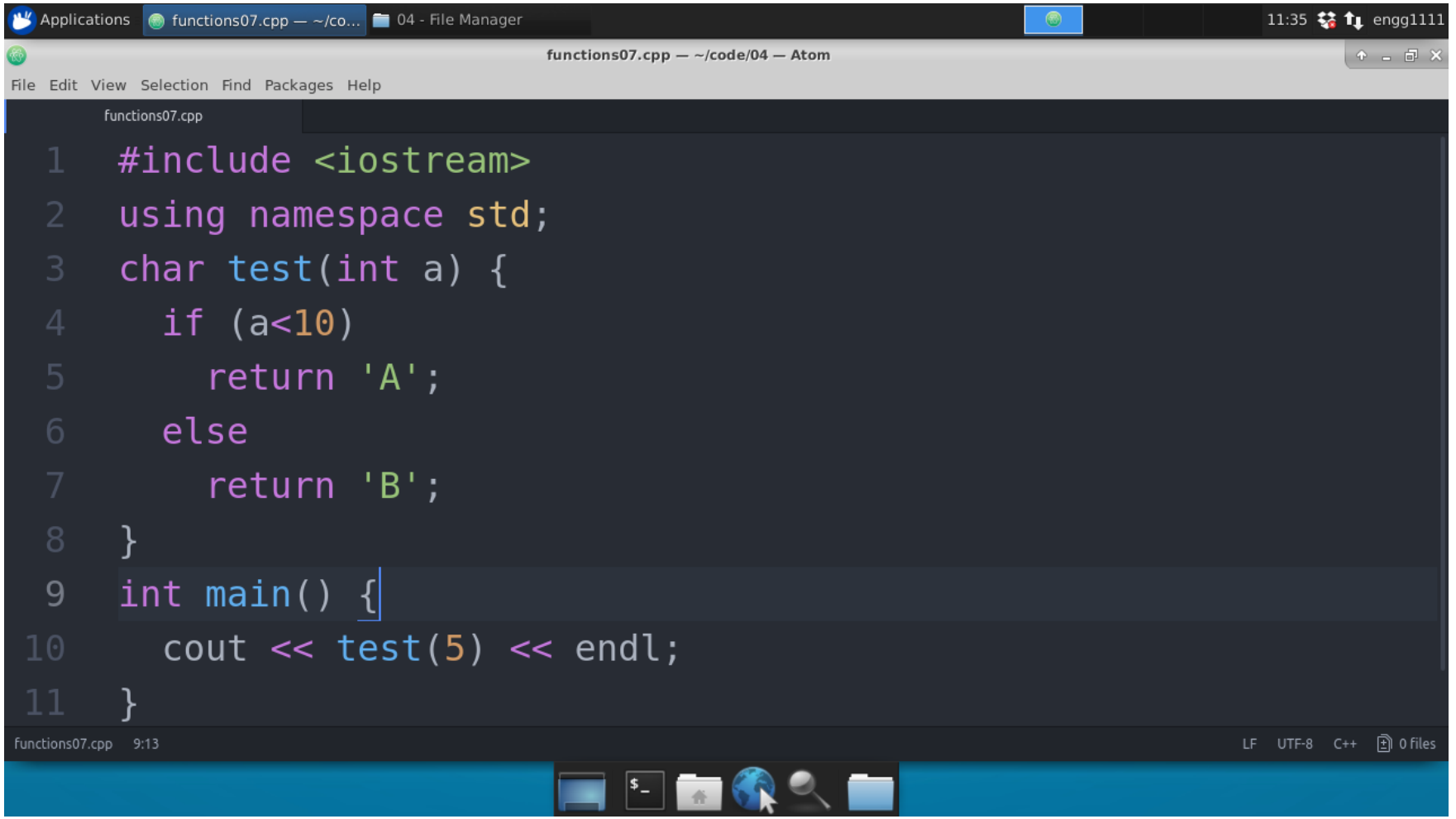
File Edit View Selection Find Packages Help

functions05.cpp

```cpp
#include <iostream>
using namespace std;
int test(int a, int b, int c) {
    cout << a <<" "<< b <<" "<< c << endl;
    return a+b+c;
}
int main() {
    cout << test(1, 3, 2) << endl;
}
```

Task

- What is the output of the following program?



Arguments

- The arguments used in a function call can be constants, variables, expressions, or even function calls

functions06.cpp — ~/code/04 — Atom

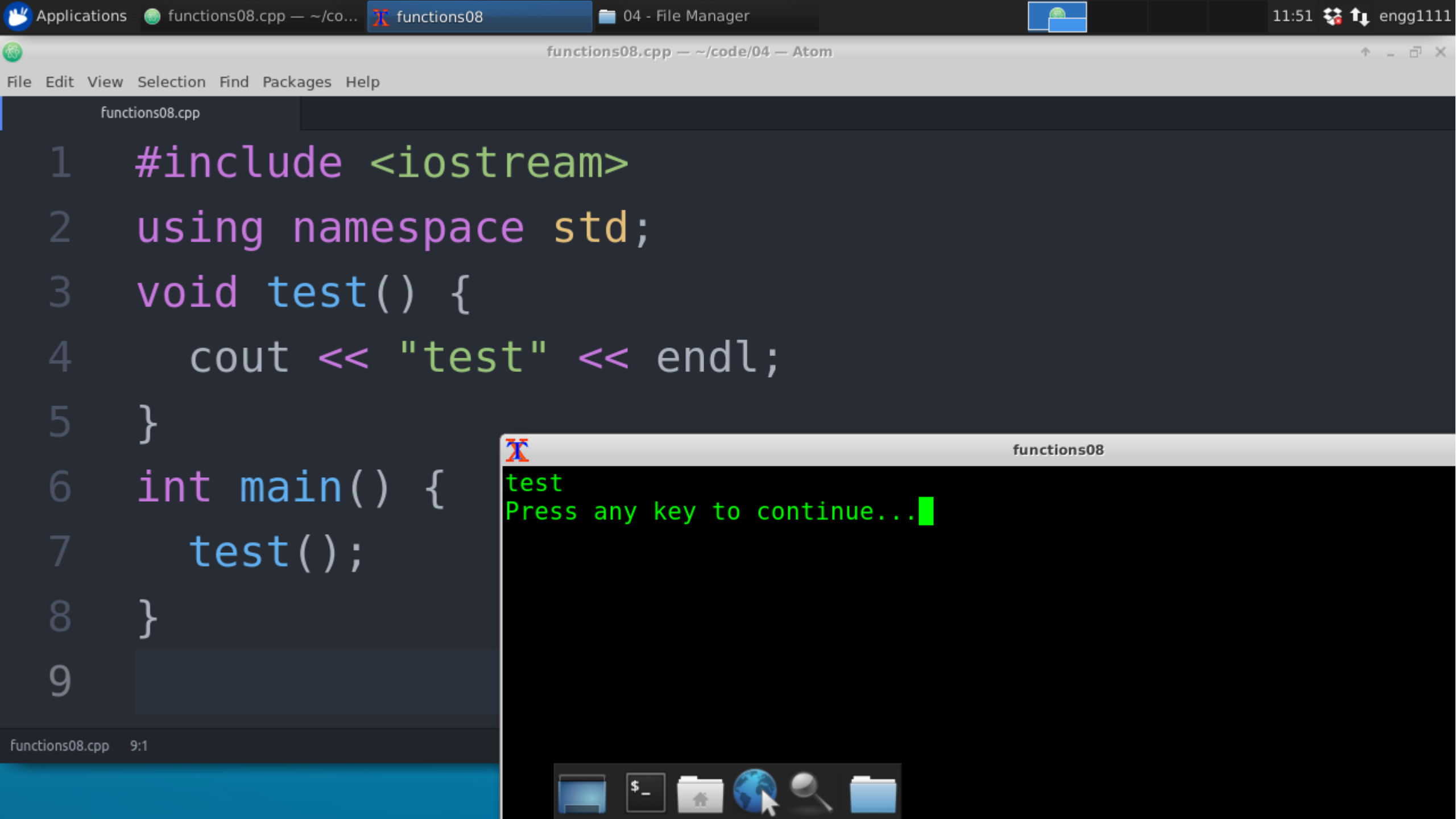
File Edit View Selection Find Packages Help

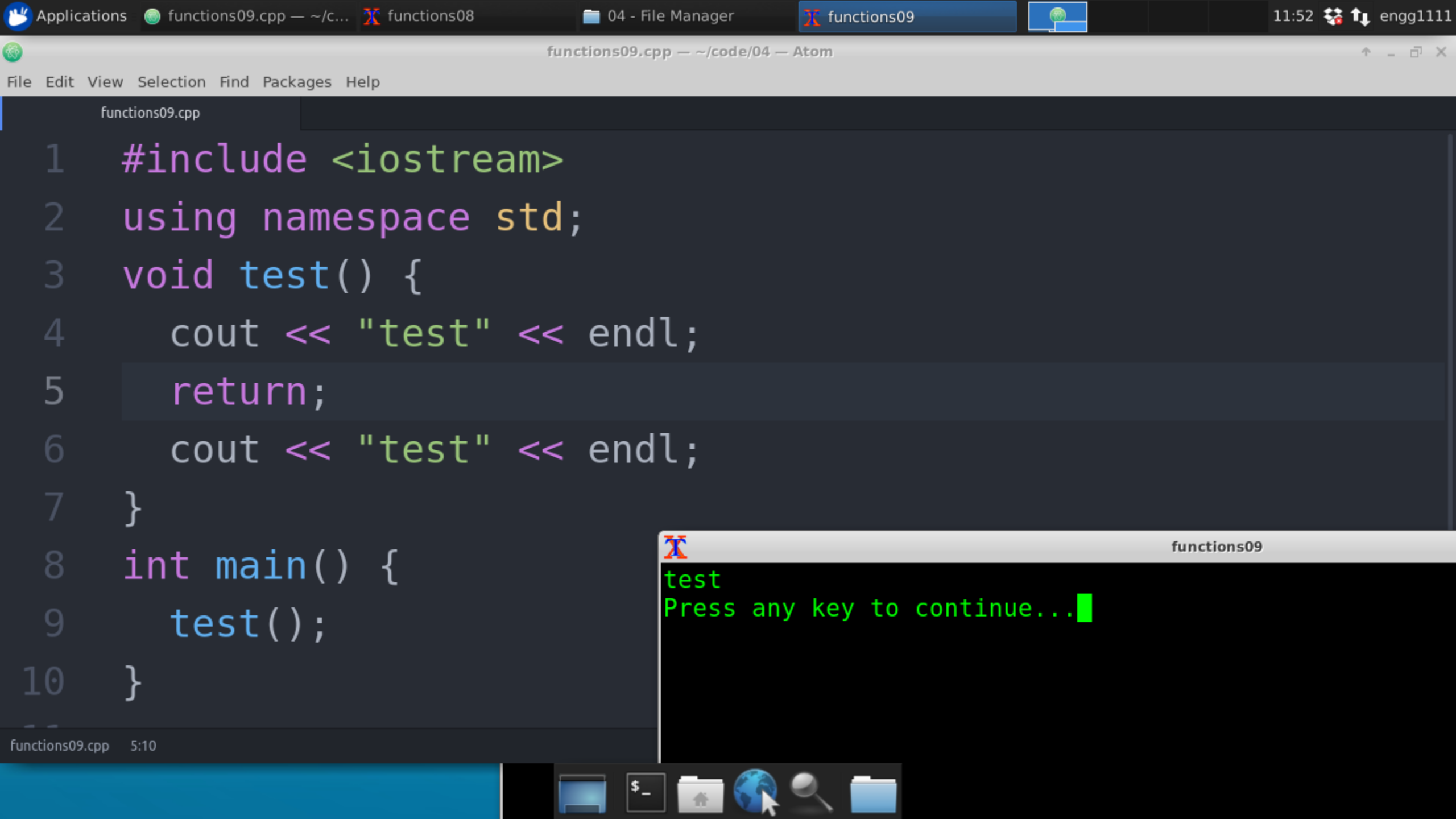
functions06.cpp

```cpp
#include <iostream>
using namespace std;
int test(int a, int b, int c) {
  cout << a <<" "<< b <<" "<< c << endl;
  return a+b+c;
}
int main() {
  cout << test(test(1, 1, 1), 3-1, 2) << endl;
}
```

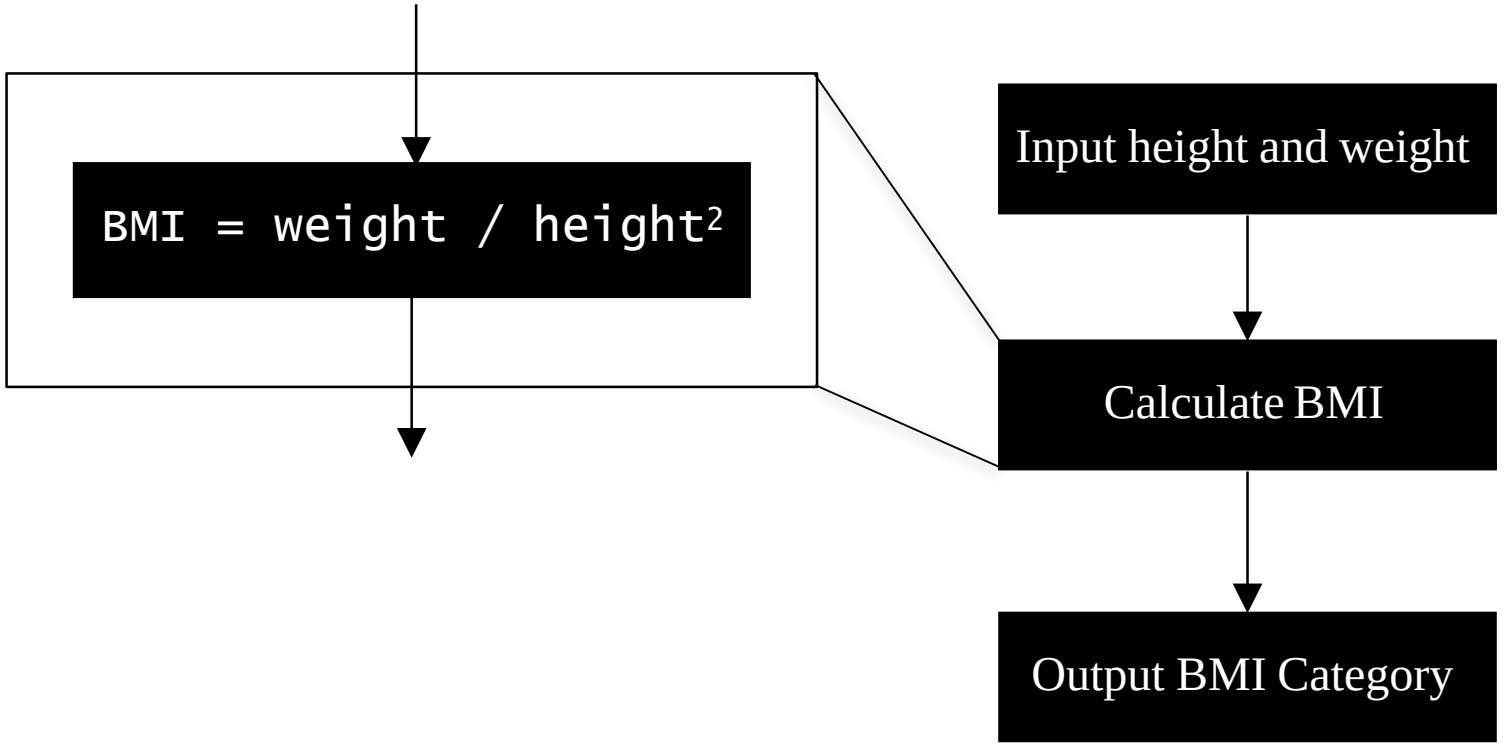
void Functions

- In some situations, a function returns no value
- In this case, we use the void type specifier
- A function with no return value is called a void function
- The return statement in a void function does not specify any return value
  - It is used to return the control to the calling function
- If a return statement is missing in a void function, the control will be returned to the calling function after the execution of the last statement in the function

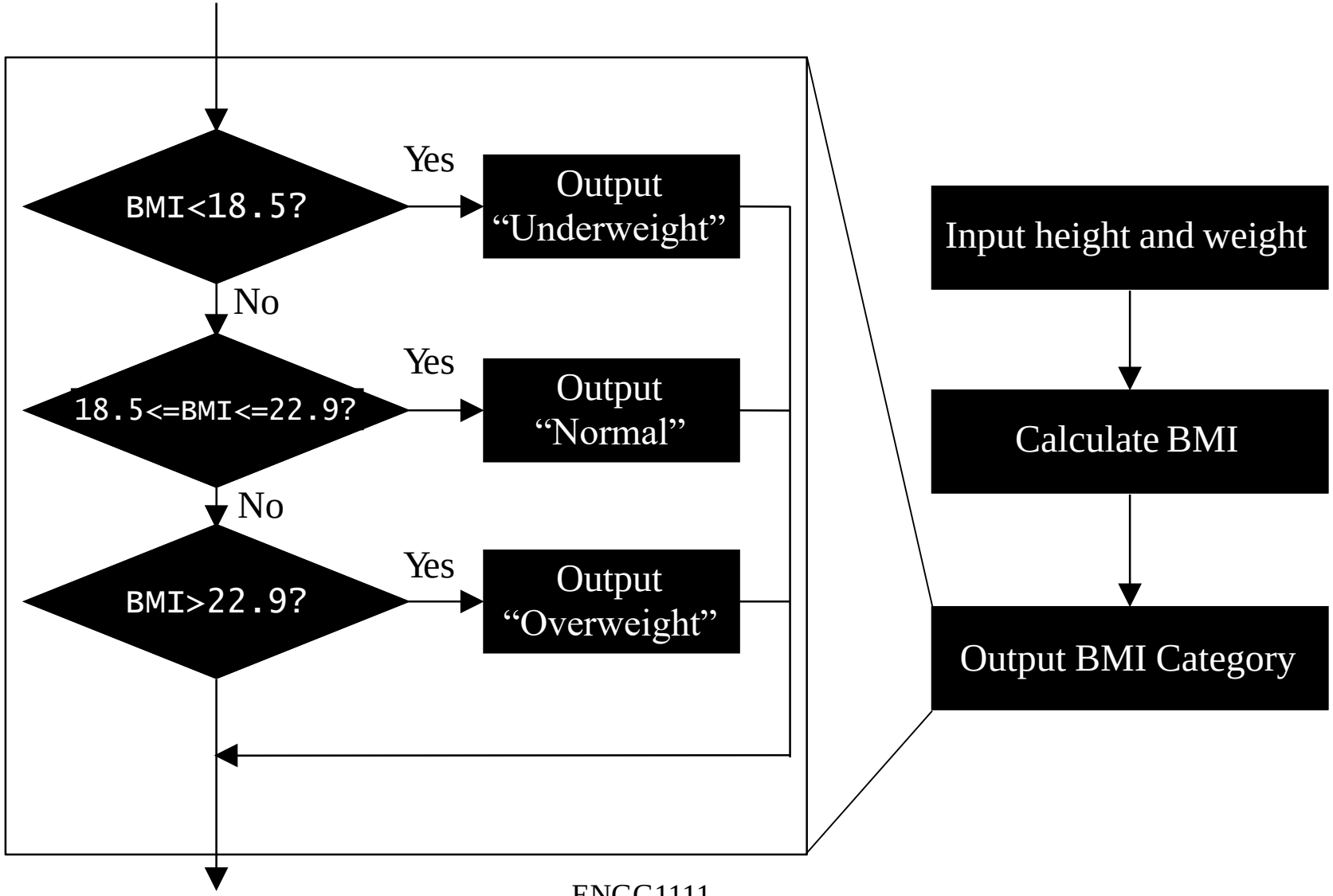

Applications
functions08.cpp — ~/co...
functions08
04 - File Manager
11:51
engg1111
functions08.cpp — ~/code/04 — Atom
File Edit View Selection Find Packages Help
functions08.cpp
1 #include <iostream>
2 using namespace std;
3 void test() {
4 cout << "test" << endl;
5 }
6 int main() {
7 test();
8 }
9
functions08.cpp 9:1
functions08
test
Press any key to continue...


Applications
functions09.cpp — ~/c...
functions08
04 - File Manager
functions09
11:52
engg1111
functions09.cpp — ~/code/04 — Atom
File Edit View Selection Find Packages Help
functions09.cpp
1 #include <iostream>
2 using namespace std;
3 void test() {
4 cout << "test" << endl;
5 return;
6 cout << "test" << endl;
7 }
8 int main() {
9 test();
10 }
functions09.cpp 5:10
functions09
test
Press any key to continue...

calculateBMI()



outputCategory()



functions10.cpp — ~/code/04 — Atom

File Edit View Selection Find Packages Help

functions10.cpp

```cpp
#include <iostream>
using namespace std;
double calculateBMI(double height, double weight) {
  return weight / (height*height);
}
void outputCategory(double bmi) {
  if (bmi<18.5)
    cout << "Underweight" << endl;
  if (bmi >= 18.5 && bmi <= 22.9)
    cout << "Normal" << endl;
  if (bmi > 22.9)
    cout << "Overweight" << endl;
}
int main() {
  double height, weight;
  cin >> height >> weight;
  double bmi = calculateBMI(height, weight);
  outputCategory(bmi);
}
```

Function Definition

- A function must be defined before the function call is made

functions11.cpp — ~/code/04 — Atom

File Edit View Selection Find Packages Help

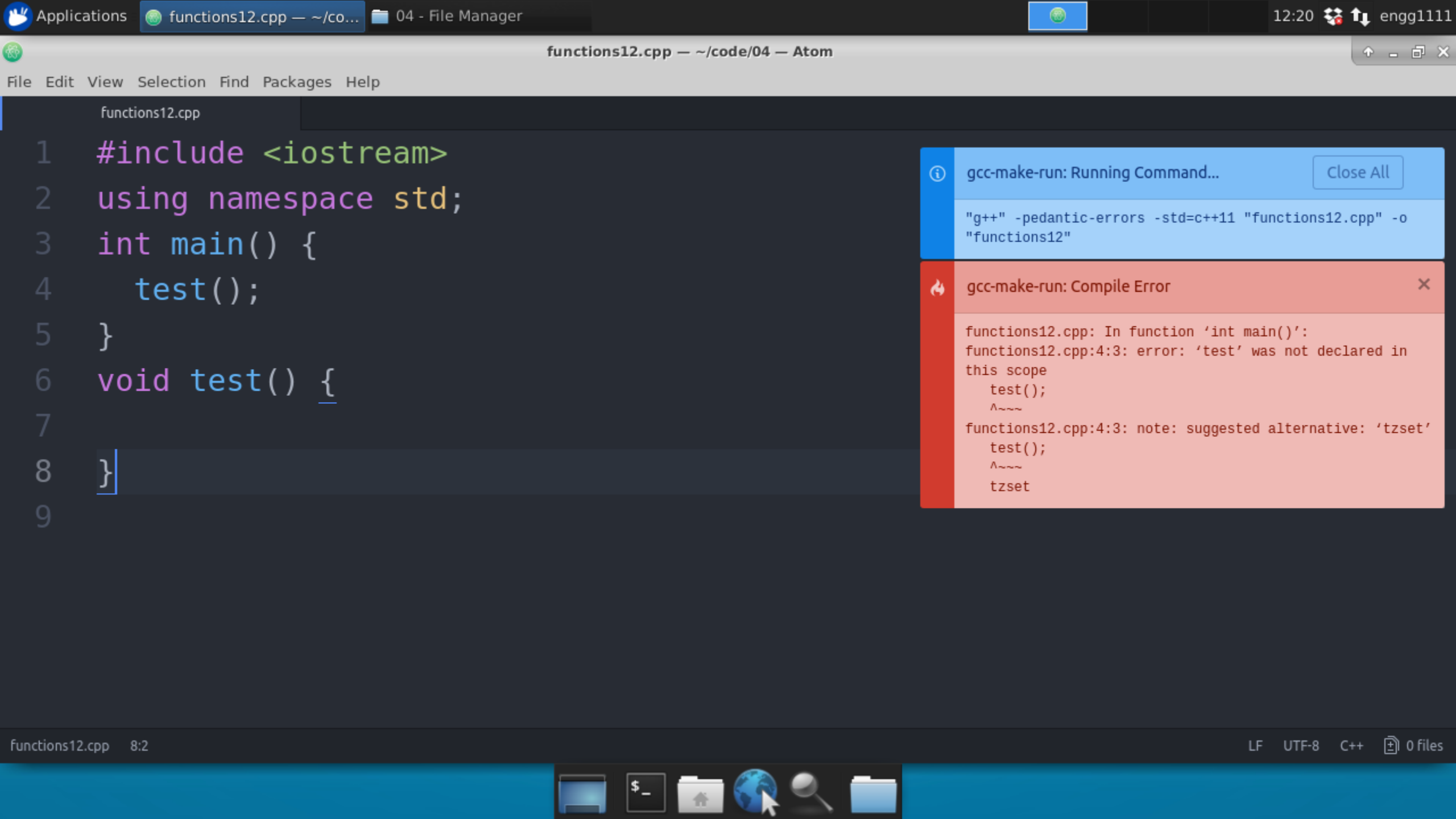
functions11.cpp

```
1  #include <iostream>
2  using namespace std;
3  void test() {
4
5  }
6  int main() {
7    test();
8  }
9
```

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

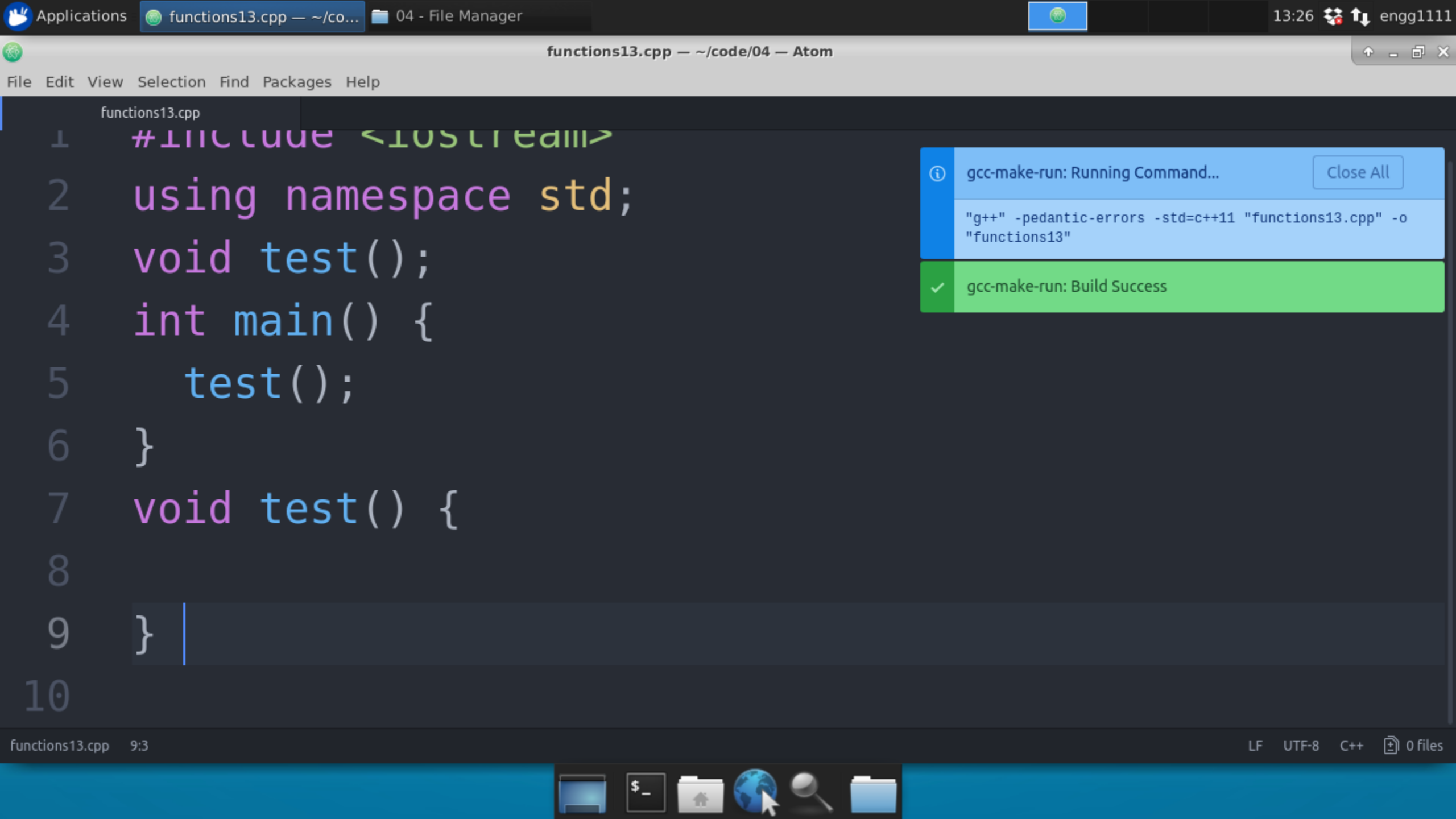
```
"g++" -pedantic-errors -std=c++11 "functions11.cpp" -o "functions11"
```

gcc-make-run: Build Success


Applications
functions12.cpp — ~/co...
04 - File Manager
12:20
engg1111
functions12.cpp — ~/code/04 — Atom
File Edit View Selection Find Packages Help
functions12.cpp
1 #include <iostream>
2 using namespace std;
3 int main() {
4 test();
5 }
6 void test() {
7
8 }
9
gcc-make-run: Running Command...
Close All
"g++" -pedantic-errors -std=c++11 "functions12.cpp" -o "functions12"
gcc-make-run: Compile Error
functions12.cpp: In function ‘int main()’:
functions12.cpp:4:3: error: ‘test’ was not declared in this scope
test();
^~~~
functions12.cpp:4:3: note: suggested alternative: ‘tzset’
test();
^~~~
tzset
functions12.cpp 8:2
LF UTF-8 C++ 0 files

Function Definition

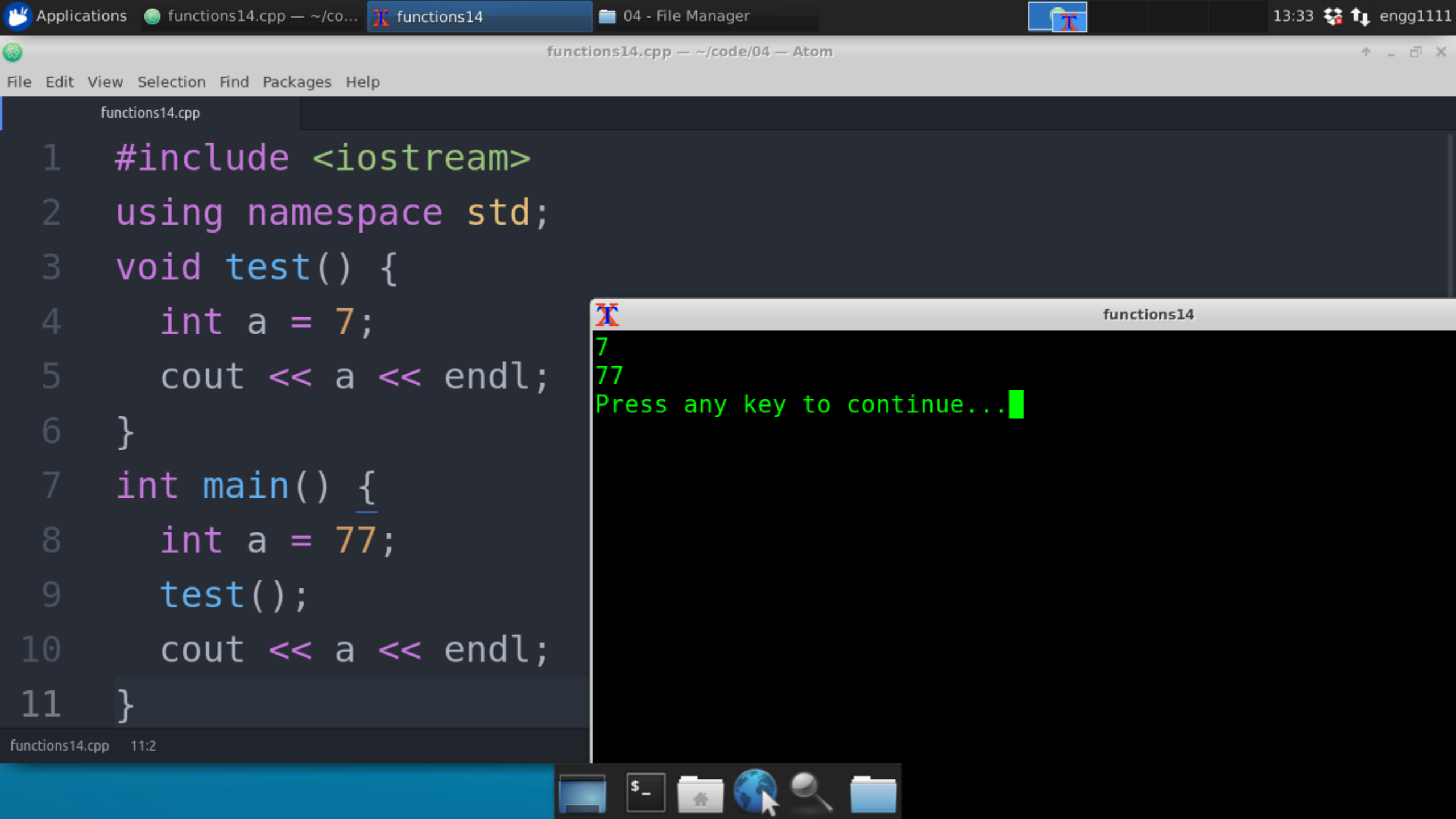
- A function must be defined before the function call is made
  - Solution 1
    - Place the function definition before the function call in the source file
  - Solution 2
    - The function definition can be placed anywhere in the source file by including just the function declaration before the function call


Applications
functions13.cpp — ~/co...
04 - File Manager
13:26
engg1111
functions13.cpp — ~/code/04 — Atom
File Edit View Selection Find Packages Help
functions13.cpp
#include <iostream>
using namespace std;
void test();
int main() {
test();
}
void test() {
}
gcc-make-run: Running Command...
Close All
"g++" -pedantic-errors -std=c++11 "functions13.cpp" -o "functions13"
gcc-make-run: Build Success
functions13.cpp 9:3
LF UTF-8 C++ 0 files

Local and Global Variables

Local Variables

- Variables declared within a function, including parameters, are local to that particular function
  - No other function can have access to them
- Local variables in a function come into existence only when the function is called, and disappear when the function is exited
- Local variables declared within the same function must have unique identifiers, whereas local variables of different functions may use the same identifier


Applications
functions14.cpp — ~/co...
functions14
04 - File Manager
13:33
engg1111
functions14.cpp — ~/code/04 — Atom
File Edit View Selection Find Packages Help
functions14.cpp
1 #include <iostream>
2 using namespace std;
3 void test() {
4 int a = 7;
5 cout << a << endl;
6 }
7 int main() {
8 int a = 77;
9 test();
10 cout << a << endl;
11 }
functions14.cpp 11:2
functions14
7
77
Press any key to continue...

Global Variables

- Variables may also be declared outside all functions
  - Such variables are called global variables
    - They can be accessed by all functions
- Global variables remain in existence permanently
  - Retain their values even after the functions that set their values have returned
- Could be used instead of arguments to communicate data between functions
- **You must avoid using global variables**
  - Since the values of global variables can be changed by several functions they are very hard to trace

functions15.cpp — ~/code/04 — Atom

File Edit View Selection Find Packages Help

functions15.cpp

```
1  #include <iostream>
2  using namespace std;
3  int g = 1;
4  void doSomething() {
5    g = 2;
6  }
7  int main() {
8    g = 1;
9    doSomething();
10   // Programmer expects g to be 1
11   // doSomething() changed it to 2
12   if (g == 1) cout << "No threats detected" << endl;
13   else cout << "Launch nuclear missiles" << endl;
14 }
```

functions15

```
Launch nuclear missiles
Press any key to continue...
```

functions16.cpp — ~/code/04 — Atom

File Edit View Selection Find Packages Help

functions16.cpp

```
1  #include <iostream>
2  using namespace std;
3  int g = 1;
4  void doSomething() {
5    int g = 2;
6  }
7  int main() {
8    g = 1;
9    doSomething();
10   // Programmer expects g to be 1
11   // doSomething() changed it to 2
12   if (g == 1) cout << "No threats detected" << endl;
13   else cout << "Launch nuclear missiles" << endl;
14 }
```

Scope of Variables

Scope of Variables

- The scope of a variable is the portion of a program in which the variable can be used
  - A variable cannot be used beyond its scope
- The scope of a local variable starts from its declaration up to the end of the block
  - A block is delimited by a pair of braces { }
  - Variables declared in outer blocks can be referred to in an inner block
- Variables can be declared with the same identifier as long as they have different scopes
  - Variables in an inner block will hide any identically named variables in outer blocks

functions17.cpp — ~/code/04 — Atom

File Edit View Selection Find Packages Help

functions17.cpp

```
1  #include <iostream>
2  using namespace std;
3  int a;
4  void test() {
5    {
6      int b;
7    }
8    int c;
9  }
10 int main() {
11   int d;
12   {
13     int e;
14   }
15 }
```

functions18.cpp — ~/code/04 — Atom

File Edit View Selection Find Packages Help

functions18.cpp

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4    int i = 0;
5    cout << "Outer block: i = " << i << endl;
6    {
7      int i = 100;
8      cout << "Inner block: i = " << i << endl;
9    }
10   cout << "Outer block: i = " << i << endl;
11 }
```

Pass by Value

- When a function call takes place, the values of the arguments are copied to the parameters of the function
- This mechanism of parameter-passing is known as pass by value

Task

- Write a function that takes two integers as arguments and swaps them
  - The function should be named swap
- Example
  - The following program, after completion, should output 21

```
1   #include <iostream>
2   using namespace std;
3   void swap(int x, int y) {
4     //your code here!
5   }
6   int main() {
7     int x = 1, y = 2;
8     swap(x,y);
9     cout << x << y << endl;
10  }
```

Applications | functions20.cpp — ~/Dr... | functions20 | 13:56 | engg1111

functions20.cpp — ~/Dropbox/VM/04 — Atom

File Edit View Selection Find Packages Help

functions20.cpp

```
1  #include <iostream>
2  using namespace std;
3  void swap(int x, int y) {
4    int tmp = x;
5    x = y;
6    y = tmp;
7  }
8  int main() {
9    int x = 1, y = 2;
10   swap(x,y);
11   cout << x << y << endl;
12 }
```

functions20.cpp 5:9

functions20

```
12
Press any key to continue...
```

Pass by Value - Analysis

```
1   #include <iostream>
2   using namespace std;
3   void swap(int x, int y) {
4     int tmp = x;
5     x = y;
6     y = tmp;
7   }
8   int main() {
9     int x = 1, y = 2;
10    swap(x,y);
11    cout << x << y << endl;
12  }
```

Pass by Value - Analysis

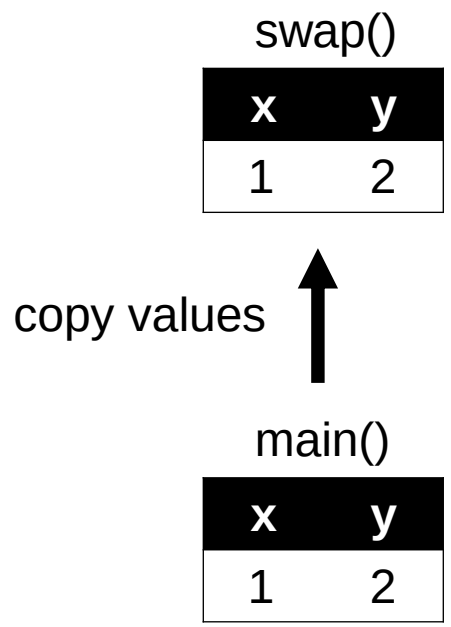
```
1   #include <iostream>
2   using namespace std;
3   void swap(int x, int y) {
4     int tmp = x;
5     x = y;
6     y = tmp;
7   }
8   int main() {
9     int x = 1, y = 2;
10    swap(x,y);
11    cout << x << y << endl;
12  }
```

main()

x	y
1	2

Pass by Value - Analysis

```
1   #include <iostream>
2   using namespace std;
3   void swap(int x, int y) {
4     int tmp = x;
5     x = y;
6     y = tmp;
7   }
8   int main() {
9     int x = 1, y = 2;
10    swap(x,y);
11    cout << x << y << endl;
12  }
```



Pass by Value - Analysis

```
1  #include <iostream>
2  using namespace std;
3  void swap(int x, int y) {
4    int tmp = x;
5    x = y;
6    y = tmp;
7  }
8  int main() {
9    int x = 1, y = 2;
10   swap(x,y);
11   cout << x << y << endl;
12 }
```

swap()

x	y	tmp
1	2	1

main()

x	y
1	2

Pass by Value - Analysis

```
1  #include <iostream>
2  using namespace std;
3  void swap(int x, int y) {
4    int tmp = x;
5    x = y;
6    y = tmp;
7  }
8  int main() {
9    int x = 1, y = 2;
10   swap(x,y);
11   cout << x << y << endl;
12 }
```

swap()

x	y	tmp
2	2	1

main()

x	y
1	2

Pass by Value - Analysis

```
1   #include <iostream>
2   using namespace std;
3   void swap(int x, int y) {
4     int tmp = x;
5     x = y;
6     y = tmp;
7   }
8   int main() {
9     int x = 1, y = 2;
10    swap(x,y);
11    cout << x << y << endl;
12  }
```

swap()

x	y	tmp
2	1	1

main()

x	y
1	2

Pass by Value - Analysis

```
1   #include <iostream>
2   using namespace std;
3   void swap(int x, int y) {
4     int tmp = x;
5     x = y;
6     y = tmp;
7   }
8   int main() {
9     int x = 1, y = 2;
10    swap(x,y);
11    cout << x << y << endl;
12  }
```

main()

x	y
1	2

Pass by Value - Analysis

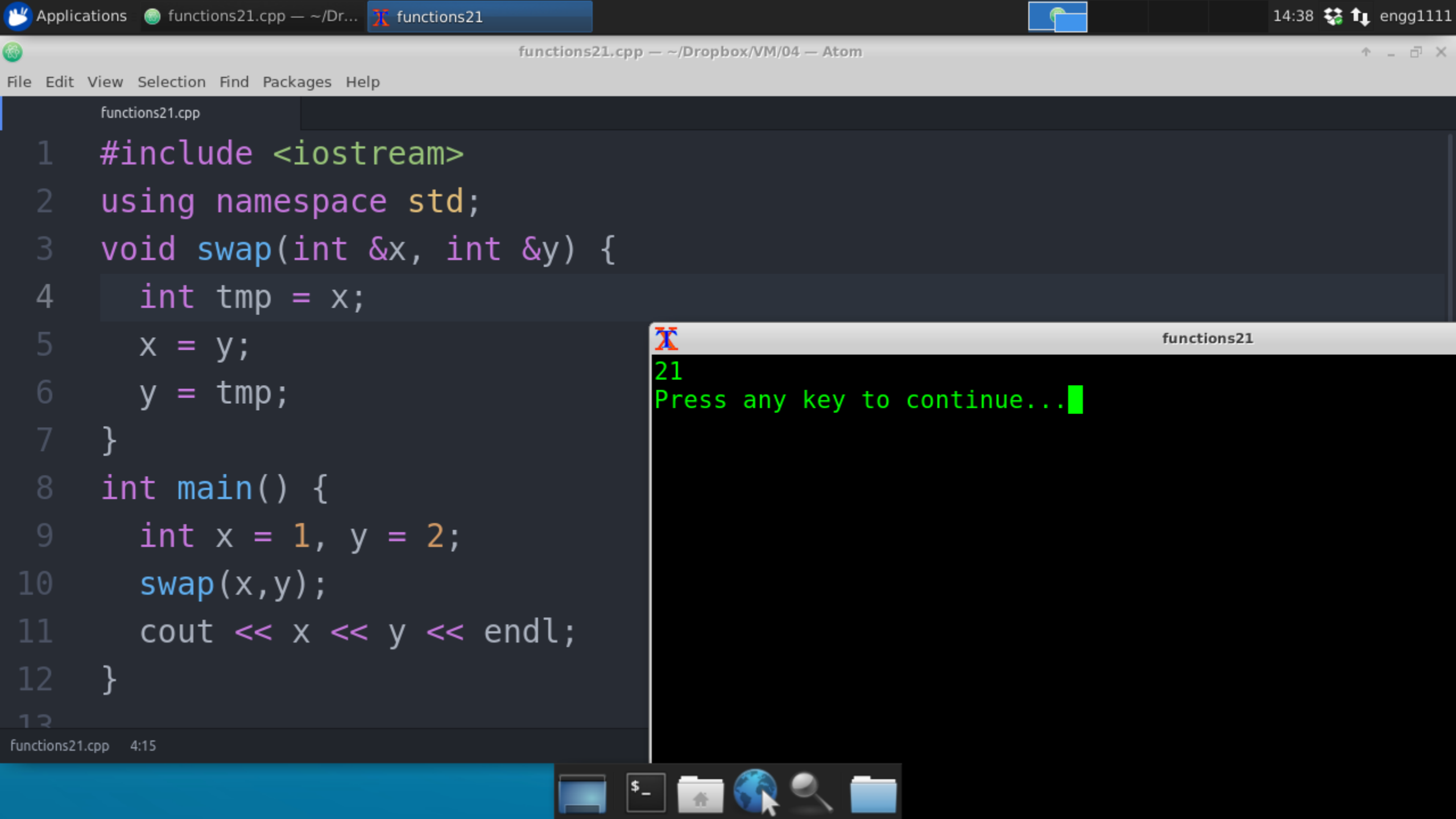
```
1  #include <iostream>
2  using namespace std;
3  void swap(int x, int y) {
4    int tmp = x;
5    x = y;
6    y = tmp;
7  }
8  int main() {
9    int x = 1, y = 2;
10   swap(x,y);
11   cout << x << y << endl;
12 }
```

main()

x	y
1	2

Pass by Reference

- In order to allow a function to modify the variables in the calling function, another parameter-passing mechanism known as pass by reference should be used
- In pass by reference the function parameters will refer to the same variable


Applications
functions21.cpp — ~/Dr...
functions21
14:38
engg1111
functions21.cpp — ~/Dropbox/VM/04 — Atom
File Edit View Selection Find Packages Help
functions21.cpp
1 #include <iostream>
2 using namespace std;
3 void swap(int &x, int &y) {
4 int tmp = x;
5 x = y;
6 y = tmp;
7 }
8 int main() {
9 int x = 1, y = 2;
10 swap(x,y);
11 cout << x << y << endl;
12 }
functions21.cpp 4:15
functions21
21
Press any key to continue...

Pass by Reference - Analysis

```
1   #include <iostream>
2   using namespace std;
3   void swap(int &x, int &y) {
4     int tmp = x;
5     x = y;
6     y = tmp;
7   }
8   int main() {
9     int x = 1, y = 2;
10    swap(x,y);
11    cout << x << y << endl;
12  }
```

Pass by Reference - Analysis

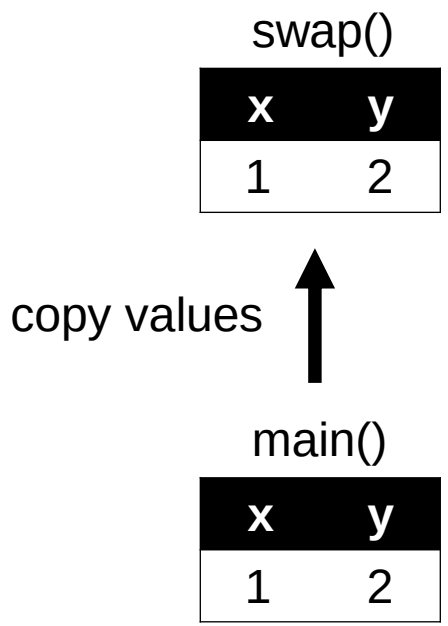
```
1   #include <iostream>
2   using namespace std;
3   void swap(int &x, int &y) {
4     int tmp = x;
5     x = y;
6     y = tmp;
7   }
8   int main() {
9     int x = 1, y = 2;
10    swap(x,y);
11    cout << x << y << endl;
12  }
```

main()

x	y
1	2

Pass by Reference - Analysis

```
1   #include <iostream>
2   using namespace std;
3   void swap(int &x, int &y) {
4     int tmp = x;
5     x = y;
6     y = tmp;
7   }
8   int main() {
9     int x = 1, y = 2;
10    swap(x,y);
11    cout << x << y << endl;
12  }
```



Pass by Reference - Analysis

```
1   #include <iostream>
2   using namespace std;
3   void swap(int &x, int &y) {
4     int tmp = x;
5     x = y;
6     y = tmp;
7   }
8   int main() {
9     int x = 1, y = 2;
10    swap(x,y);
11    cout << x << y << endl;
12  }
```

swap()

x	y	tmp
1	2	1

main()

x	y
1	2

Pass by Reference - Analysis

```
1   #include <iostream>
2   using namespace std;
3   void swap(int &x, int &y) {
4     int tmp = x;
5     x = y;
6     y = tmp;
7   }
8   int main() {
9     int x = 1, y = 2;
10    swap(x,y);
11    cout << x << y << endl;
12  }
```

swap()

x	y	tmp
2	2	1

main()

x	y
2	2

Pass by Reference - Analysis

```
1   #include <iostream>
2   using namespace std;
3   void swap(int &x, int &y) {
4     int tmp = x;
5     x = y;
6     y = tmp;
7   }
8   int main() {
9     int x = 1, y = 2;
10    swap(x,y);
11    cout << x << y << endl;
12  }
```

swap()

x	y	tmp
2	1	1

main()

x	y
2	1

Pass by Reference - Analysis

```
1   #include <iostream>
2   using namespace std;
3   void swap(int &x, int &y) {
4     int tmp = x;
5     x = y;
6     y = tmp;
7   }
8   int main() {
9     int x = 1, y = 2;
10    swap(x,y);
11    cout << x << y << endl;
12  }
```

main()

x	y
2	1

Pass by Reference - Analysis

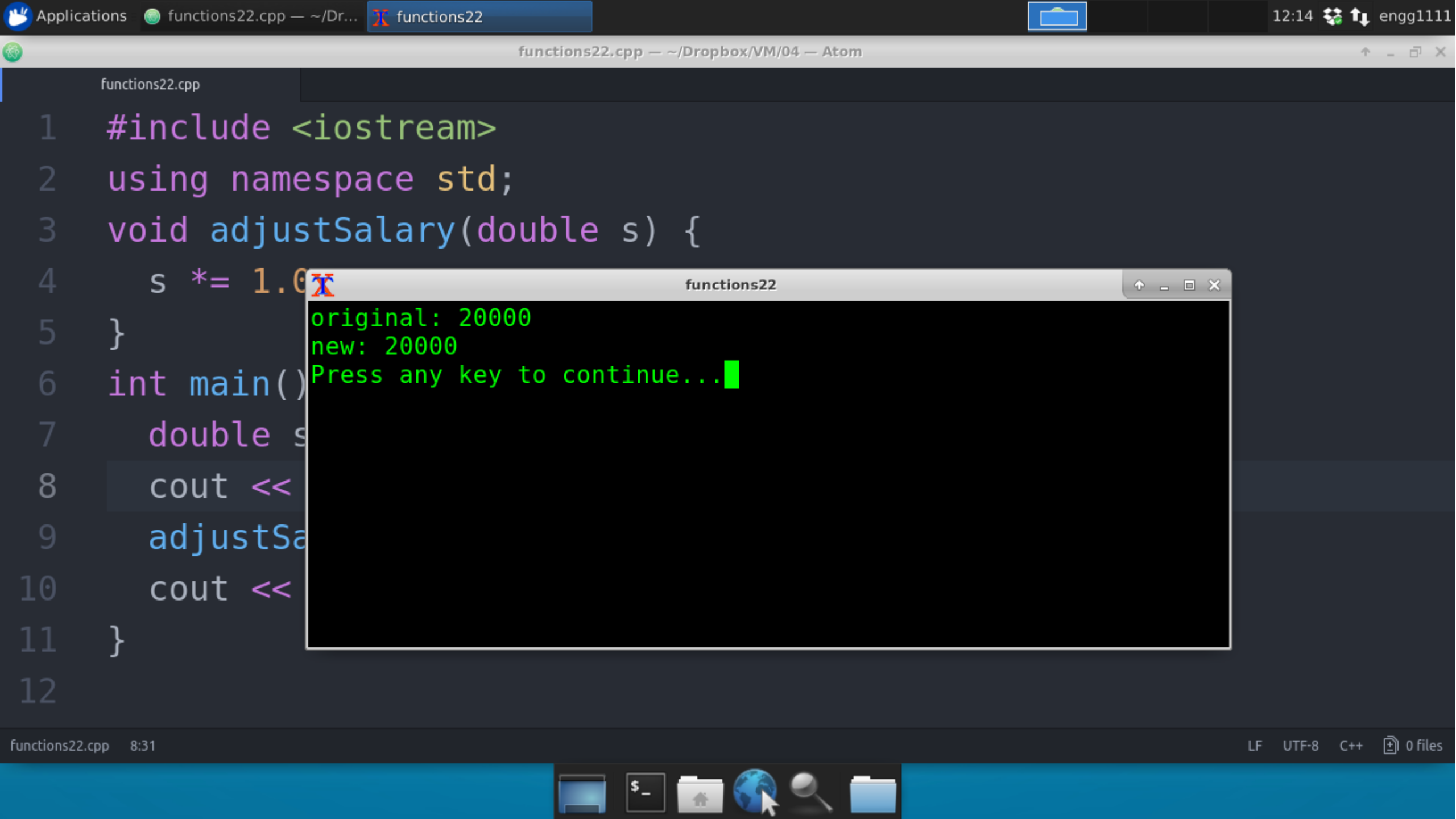
```
1   #include <iostream>
2   using namespace std;
3   void swap(int &x, int &y) {
4     int tmp = x;
5     x = y;
6     y = tmp;
7   }
8   int main() {
9     int x = 1, y = 2;
10    swap(x,y);
11    cout << x << y << endl;
12  }
```

main()

x	y
2	1

functions22.cpp

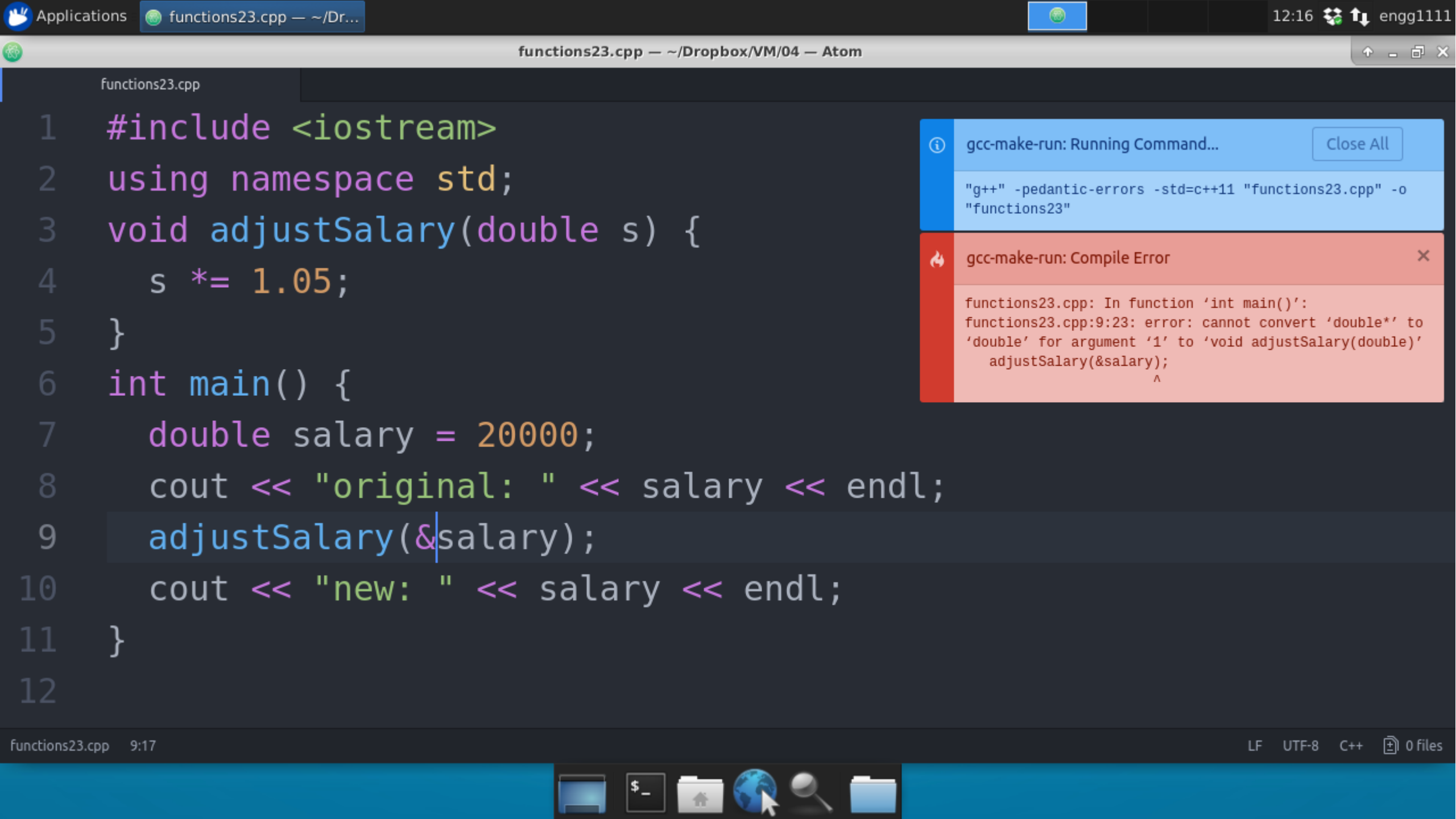
```cpp
#include <iostream>
using namespace std;
void adjustSalary(double s) {
  s *= 1.05;
}
int main() {
  double salary = 20000;
  cout << "original: " << salary << endl;
  adjustSalary(salary);
  cout << "new: " << salary << endl;
}

```


Applications
functions22.cpp — ~/Dr...
functions22
12:14
engg1111
functions22.cpp — ~/Dropbox/VM/04 — Atom
functions22.cpp
1 #include <iostream>
2 using namespace std;
3 void adjustSalary(double s) {
4 s *= 1.0
5 }
6 int main()
7 double s
8 cout <<
9 adjustSa
10 cout <<
11 }
12
functions22
original: 20000
new: 20000
Press any key to continue...
functions22.cpp 8:31
LF UTF-8 C++ 0 files

functions23.cpp

```
1  #include <iostream>
2  using namespace std;
3  void adjustSalary(double s) {
4    s *= 1.05;
5  }
6  int main() {
7    double salary = 20000;
8    cout << "original: " << salary << endl;
9    adjustSalary(&salary);
10   cout << "new: " << salary << endl;
11 }
12
```

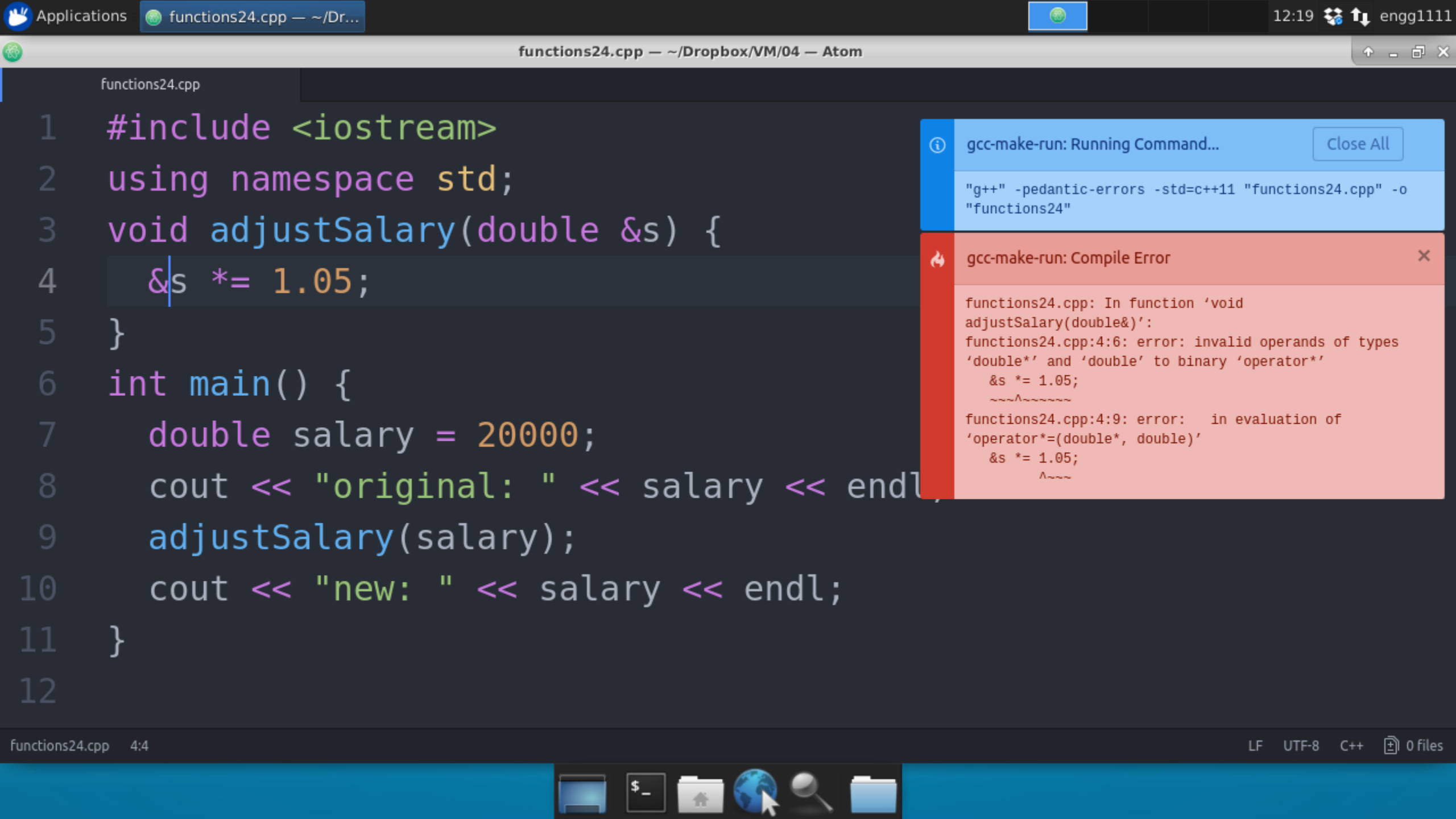
Proposed Fix 1


Applications
functions23.cpp — ~/Dr...
12:16
engg1111
functions23.cpp — ~/Dropbox/VM/04 — Atom
functions23.cpp
#include <iostream>
using namespace std;
void adjustSalary(double s) {
s *= 1.05;
}
int main() {
double salary = 20000;
cout << "original: " << salary << endl;
adjustSalary(&salary);
cout << "new: " << salary << endl;
}
gcc-make-run: Running Command...
Close All
"g++" -pedantic-errors -std=c++11 "functions23.cpp" -o "functions23"
gcc-make-run: Compile Error
functions23.cpp: In function 'int main()':
functions23.cpp:9:23: error: cannot convert 'double*' to 'double' for argument '1' to 'void adjustSalary(double)'
adjustSalary(&salary);
^
functions23.cpp 9:17
LF UTF-8 C++ 0 files

functions24.cpp

```
1  #include <iostream>
2  using namespace std;
3  void adjustSalary(double &s) {
4    &s *= 1.05;
5  }
6  int main() {
7    double salary = 20000;
8    cout << "original: " << salary << endl;
9    adjustSalary(salary);
10   cout << "new: " << salary << endl;
11 }
12
```

Proposed Fix 2


Applications
functions24.cpp — ~/Dr...
12:19
engg1111
functions24.cpp — ~/Dropbox/VM/04 — Atom
functions24.cpp
#include <iostream>
using namespace std;
void adjustSalary(double &s) {
&s *= 1.05;
}
int main() {
double salary = 20000;
cout << "original: " << salary << endl;
adjustSalary(salary);
cout << "new: " << salary << endl;
}
gcc-make-run: Running Command...
Close All
"g++" -pedantic-errors -std=c++11 "functions24.cpp" -o "functions24"
gcc-make-run: Compile Error
functions24.cpp: In function ‘void adjustSalary(double&)’:
functions24.cpp:4:6: error: invalid operands of types ‘double*’ and ‘double’ to binary ‘operator*’
&s *= 1.05;
~~~^~~~~~
functions24.cpp:4:9: error: in evaluation of ‘operator*=(double*, double)’
&s *= 1.05;
^~~~
functions24.cpp 4:4
LF
UTF-8
C++
0 files
~~~

Applications | functions24.cpp — ~/Dr... | functions24 | 12:20 | engg1111

functions24.cpp — ~/Dropbox/VM/04 — Atom

functions24.cpp

```cpp
#include <iostream>
using namespace std;
void adjustSalary(double &s) {
  s *= 1.05;
}
int main() {
  double salary = 20000;
  cout << "original: " << salar
  adjustSalary(salary);
  cout << "new: " << salary <<
}

```

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

```
"g++" -pedantic-errors -std=c++11 "functions24.cpp" -o "functions24"
```

gcc-make-run: Build Success

functions24

```
original: 20000
new: 21000
Press any key to continue...
```

functions24.cpp 3:32

Task

- If you understand the concepts of pass by value and pass by reference, you should be able to work out the output of this program

```
1  #include <iostream>
2  using namespace std;
3  void outputXY(int x, int y) {
4    cout << x << y << endl;
5  }
6  void swap(int x, int &y) {
7    outputXY(x,y);
8    int tmp = x;
9    x = y;
10   y = tmp;
11   outputXY(x,y);
12 }
13 int main() {
14   int x = 0, y = 4;
15   outputXY(x, y);
16   swap(y,x);
17   if (x<5) swap (x,y);
18   outputXY(x,y);
19 }
```

```
1  #include <iostream>
2  using namespace std;
3  void outputXY(int x, int y) {
4    cout << x << y << endl;
5  }
6  void swap(int x, int &y) {
7    outputXY(x,y);
8    int tmp = x;
9    x = y;
10   y = tmp;
11   outputXY(x,y);
12 }
13 int main() {
14   int x = 0, y = 4;
15   outputXY(x, y);
16   swap(y,x);
17   if (x<5) swap (x,y);
18   outputXY(x,y);
19 }
```

```
1  #include <iostream>
2  using namespace std;
3  void outputXY(int x, int y) {
4    cout << x << y << endl;
5  }
6  void swap(int x, int &y) {
7    outputXY(x,y);
8    int tmp = x;
9    x = y;
10   y = tmp;
11   outputXY(x,y);
12 }
13 int main() {
14   int x = 0, y = 4;
15   outputXY(x, y);
16   swap(y,x);
17   if (x<5) swap (x,y);
18   outputXY(x,y);
19 }
```

main()

x	y
0	4

04

```
1  #include <iostream>
2  using namespace std;
3  void outputXY(int x, int y) {
4    cout << x << y << endl;
5  }
6  void swap(int x, int &y) {
7    outputXY(x,y);
8    int tmp = x;
9    x = y;
10   y = tmp;
11   outputXY(x,y);
12 }
13 int main() {
14   int x = 0, y = 4;
15   outputXY(x, y);
16   swap(y,x);
17   if (x<5) swap (x,y);
18   outputXY(x,y);
19 }
```

main()

x	y
0	4

04

```
1  #include <iostream>
2  using namespace std;
3  void outputXY(int x, int y) {
4    cout << x << y << endl;
5  }
6  void swap(int x, int &y) {
7    outputXY(x,y);
8    int tmp = x;
9    x = y;
10   y = tmp;
11   outputXY(x,y);
12 }
13 int main() {
14   int x = 0, y = 4;
15   outputXY(x, y);
16   swap(y,x);
17   if (x<5) swap (x,y);
18   outputXY(x,y);
19 }
```

swap()

x	y
0	4

main()

x	y
0	4

04
40

```
1  #include <iostream>
2  using namespace std;
3  void outputXY(int x, int y) {
4    cout << x << y << endl;
5  }
6  void swap(int x, int &y) {
7    outputXY(x,y);
8    int tmp = x;
9    x = y;
10   y = tmp;
11   outputXY(x,y);
12 }
13 int main() {
14   int x = 0, y = 4;
15   outputXY(x, y);
16   swap(y,x);
17   if (x<5) swap (x,y);
18   outputXY(x,y);
19 }
```

swap()

x	y
4	0

main()

x	y
0	4

04
40

```
1  #include <iostream>
2  using namespace std;
3  void outputXY(int x, int y) {
4    cout << x << y << endl;
5  }
6  void swap(int x, int &y) {
7    outputXY(x,y);
8    int tmp = x;
9    x = y;
10   y = tmp;
11   outputXY(x,y);
12 }
13 int main() {
14   int x = 0, y = 4;
15   outputXY(x, y);
16   swap(y,x);
17   if (x<5) swap (x,y);
18   outputXY(x,y);
19 }
```

swap()

x	y	tmp
4	0	4

main()

x	y
0	4

04
40

```
1  #include <iostream>
2  using namespace std;
3  void outputXY(int x, int y) {
4    cout << x << y << endl;
5  }
6  void swap(int x, int &y) {
7    outputXY(x,y);
8    int tmp = x;
9    x = y;
10   y = tmp;
11   outputXY(x,y);
12 }
13 int main() {
14   int x = 0, y = 4;
15   outputXY(x, y);
16   swap(y,x);
17   if (x<5) swap (x,y);
18   outputXY(x,y);
19 }
```

swap()

x	y	tmp
0	0	4

main()

x	y
0	4

04
40

```
1  #include <iostream>
2  using namespace std;
3  void outputXY(int x, int y) {
4    cout << x << y << endl;
5  }
6  void swap(int x, int &y) {
7    outputXY(x,y);
8    int tmp = x;
9    x = y;
10   y = tmp;
11   outputXY(x,y);
12 }
13 int main() {
14   int x = 0, y = 4;
15   outputXY(x, y);
16   swap(y,x);
17   if (x<5) swap (x,y);
18   outputXY(x,y);
19 }
```

swap()

x	y	tmp
0	4	4

main()

x	y
4	4

04
40
04

```
1  #include <iostream>
2  using namespace std;
3  void outputXY(int x, int y) {
4    cout << x << y << endl;
5  }
6  void swap(int x, int &y) {
7    outputXY(x,y);
8    int tmp = x;
9    x = y;
10   y = tmp;
11   outputXY(x,y);
12 }
13 int main() {
14   int x = 0, y = 4;
15   outputXY(x, y);
16   swap(y,x);
17   if (x<5) swap (x,y);
18   outputXY(x,y);
19 }
```

swap()

x	y	tmp
0	4	4

main()

x	y
4	4

04
40
04

```
1  #include <iostream>
2  using namespace std;
3  void outputXY(int x, int y) {
4    cout << x << y << endl;
5  }
6  void swap(int x, int &y) {
7    outputXY(x,y);
8    int tmp = x;
9    x = y;
10   y = tmp;
11   outputXY(x,y);
12 }
13 int main() {
14   int x = 0, y = 4;
15   outputXY(x, y);
16   swap(y,x);
17   if (x<5) swap (x,y);
18   outputXY(x,y);
19 }
```

main()

x	y
4	4

04
40
04
44
44

```
1  #include <iostream>
2  using namespace std;
3  void outputXY(int x, int y) {
4    cout << x << y << endl;
5  }
6  void swap(int x, int &y) {
7    outputXY(x,y);
8    int tmp = x;
9    x = y;
10   y = tmp;
11   outputXY(x,y);
12 }
13 int main() {
14   int x = 0, y = 4;
15   outputXY(x, y);
16   swap(y,x);
17   if (x<5) swap (x,y);
18   outputXY(x,y);
19 }
```

main()

x	y
4	4

04
40
04
44
44
44

```
1  #include <iostream>
2  using namespace std;
3  void outputXY(int x, int y) {
4    cout << x << y << endl;
5  }
6  void swap(int x, int &y) {
7    outputXY(x,y);
8    int tmp = x;
9    x = y;
10   y = tmp;
11   outputXY(x,y);
12 }
13 int main() {
14   int x = 0, y = 4;
15   outputXY(x, y);
16   swap(y,x);
17   if (x<5) swap (x,y);
18   outputXY(x,y);
19 }
```

main()

x	y
4	4

Task

- What is the output of this program?

```
1  #include <iostream>
2  using namespace std;
3  void figureMeOut(int &x, int y, int &z) {
4    cout << x << " "<< y << " " << z << endl;
5    x = 1;
6    y = 2;
7    z = 3;
8    cout << x << " "<< y << " " << z << endl;
9  }
10 int main() {
11   int a=10, b=20, c=30;
12   figureMeOut(a, b, c);
13   cout << a << " "<< b << " " << c << endl;
14 }
```