Introduction to C++ and Modern Fortran

Prof. Jeremy Roberts

ME 701 – Fall 2016

Outline

Today: Basic syntax and compiling

Next time: Basic structuring, functions, Eclipse, and other functionality

Hello World

```cpp
#include <iostream>

int main(int argc, char* argv[])
{
 /* Comments can continue
    on multiple lines */

 // or just be one-liners
 std::cout << "Hello World!"
           << std::endl;

 // Because "main" is an integer
 // function, it must return an
 // integer.
 return 0;
}
```

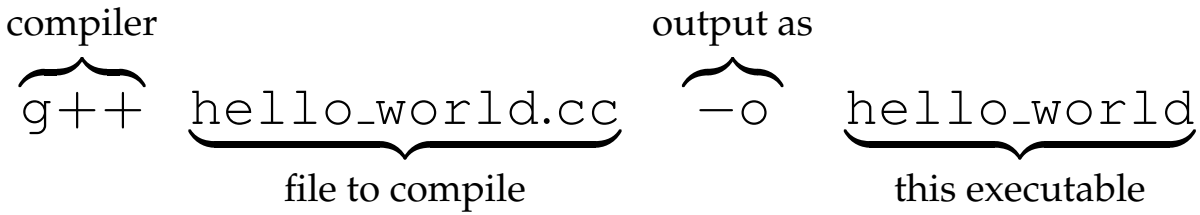
hello_world.cc

```fortran
program hello_world

 ! Fortran comments start with
 ! exclamation points, and there
 ! is not a multiline option

 print *, "Hello world!"

end program hello_world
```

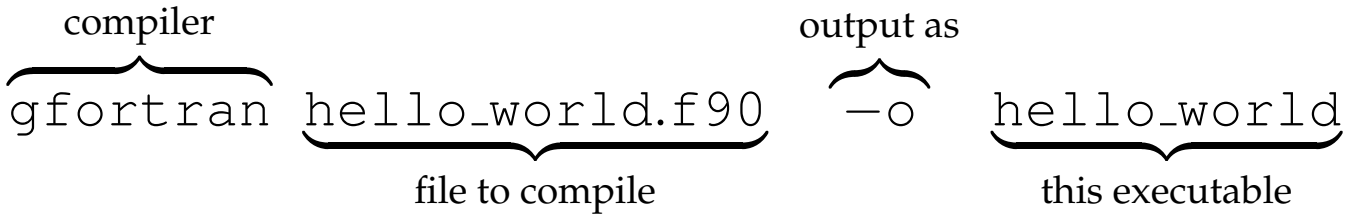
hello_world.f90

Compiling Your First Program

For C++, use (in the command line)



For Fortran, use



Use `sudo apt-get install g++ gfortran` to get them. **Now try them!**

Compiler Options

`g++` and `gfortran` are part of the GNU compiler set and share several key compiler options that may (or may not) work with compilers from other vendors; these include:

- `-Wall` – warn us of anything unexpected but make the executable
- `-Werror` – turn any warning into an error
- `-O` – (that's an "Oh") use optimization (or `-ON` for $N = 0, 1, 2, 3$ for various levels of optimization)
- `-g` – produce debugging information
- `-pg` – produce profiling information

Declaring Variables

```cpp
int main()
{
  // One can declare and then
  // define variables anywhere
  int a;
  double b;
  a = 123;
  b = 3.14;

  // One can also declare and
  // define simultaneously
  const int A = 123;
  double B = 3.14;
  float C = 3.14;

  return 0;
}
```

declaring.cc

```fortran
program declare_demo
  ! All Fortran variables must be
  ! declared before execution of
  ! statements.  These variables
  ! may be initialized, too.
  integer, parameter :: a = 123
  double precision :: b
  real :: c = 3.1415926535897932
  b = 3.1415926535897932
  print *, b
  print *, c
end program declare_demo
```

declaring.f90

Simple Math

```cpp
#include <cmath>
int main()
{
 double x = 1.0;
 double y = 2.0;
 double z;
 z = x/y;
 z = sqrt(x);
 z = exp(y);
 z = pow(x, y);
 z = M_PI; // cmath has Pi
 return 0;
}
```

simple_math.cc

```
program simple_math
  implicit none
  double precision :: x, y, z
  x = 2.0
  y = 3.0
  z = x/y
  z = sqrt(x)
  z = exp(y)
  z = x**y
  ! no built-in Pi definition
end program simple_math
```

simple_math.f90

Control of Program Flow – If's

```cpp
#include <iostream>
using std::cout;
using std::endl;
int main()
{
  int a = 1;
  if (a > 2)
  {
    // do something
  }
  else
  {
    // do something else
  }
  if (a == 1)
    a; // do something
  else if (a > 4)
    a;
  else
    a; // do somthing else
  if (a==1) cout<<"hi"<<endl;
  return 0;
}
```

control_if.cc

```
program control
  integer :: a = 1
  if (a == 1) then
    print *, "a = 1"
  else if (a == 2) then
    print *, "a = 2"
  else
    print *, "a < 1 || a > 2"
  end if
  if (a == 1) print *, "hi"
end program control
```

control_if.f90

Control of Program Flow – Switches

```cpp
#include <iostream>
using std::cout;
using std::endl;
int main()
{
  int a = 1;
  switch (a)
  {
    case 1:
      cout << "a=1" << endl;
      break;
    case 2:
      // do something
      break;
    default:
      cout << "hi" << endl;
  }
}
```

control_case.cc

```
program control
  integer :: a = 1
  select case (a)
    case (1)
      print *, "a = 1"
    case (2)
      print *, "a = 2"
    case default
      print *, "a < 1 || a > 2"
 end select
end program control
```

control_case.f90

Loops

```cpp
#include <iostream>
using namespace std;
int main()
{
  int j1 = 0;
  int j2 = 0;
  for (int i = 0; i < 10; i++)
  {
    cout << " i = " << i << endl;
    j1 = j1 + i;
    j2 += i;
  }
  int i2 = 0;
  j1 = 0;
  do
  {
    j1 += i2;
    i2++;
  }
  while(i2 < 100);
  return 0;
}
```

loops.cc

```
program loops
  integer :: i, j
  j = 0
  do i = 1, 100
    j = j + i
  end do
  i = 1
  j = 0
  do while (i < 100)
     j = j + i
     i = i + 1
  end do
end program loops
```

loops.f90

Functions

```cpp
#include <iostream>
using std::cout;
using std::endl;
int add(int a, int b)
{
  cout << "add ints" << endl;
  return a + b;
}
int add(double a, double b)
{
  cout << "add doubles" << endl;
  return a + b;
}
int main(int argc, char* argv[])
{
  cout << add(1, 2) << endl;
  cout << add(1.0, 2.0) << endl;
  return 0;
}
```

functions.cc

```
program functions
  interface add
    real function add_d(x, y)
      real, intent(in) :: x, y
    end function add_d
    integer function add_i(x, y)
      integer, intent(in) :: x, y
    end function add_i
  end interface
  print *, add(1, 2)
  print *, add(1.0, 2.0) !!!
end program functions
real function add_d(x, y)
  real, intent(in) :: x, y
  print *, "add reals"
  add_d = x + y
end function add_d
integer function add_i(x, y)
  integer, intent(in) :: x, y
  print *, "add ints"
  add_i = x + y
end function add_i
```

functions.f90

Eclipse with C++ and Fortran

- Head to `http://www.eclipse.org/downloads/`
- Click on "Download 64-Bit" button (or whatever shows up for your system), which gets you a file like `eclipse-inst-linux64.tar.gz`
- `tar -xf eclipse-inst-linux64.tar.gz`
- `cd eclipse-inst`
- If you need Java 8:
  1. `sudo add-apt-repository ppa:webupd8team/java`
  2. `sudo apt-get update`
  3. `sudo apt-get install oracle-java8-installer`
- `./eclipse-inst`
- Select Parallel Developers option (for C++ and Fortran Support)

Command Line Arguments

```cpp
#include <iostream>
#include <string>
#include <sstream>
using namespace std;
int main(int argc, char* argv[])
{
  if (argc != 2)
    {
      cout << "usage: " << argv[0]
           << " <arg>" << endl;
    }
  else
  {
    std::string s = argv[1];
    cout << "arg = " << s << endl;
    int n = 1;
    if (!(istringstream(s) >> n))
      n = 0;
    cout << "n = " << n << endl;
  }
  return 0;
}
```

command_line.cc

```
program command_line
  implicit none
  character(80) :: s
  integer :: n = 1, io
  if (command_argument_count() &
       .lt. 1) then
    stop "usage: a.out <arg>"
  else
    call get_command_argument(1, s)
    print *, "s = ", s
    read (s, *, iostat=io) n
    if (io .ne. 0) n = 0
      print *, "n = ", n
    end if
end program command_line
```

command_line.f90

File I/O

```
int main()
{
  int j1 = 0;
  int j2 = 0;
  for (int i = 0; i < 100; ++i)
  {
    j1 = j1 + i;
    j2 += i;
  }
  int i2 = 0;
  j1 = 0;
  do
  {
    j1 += i2;
    i2++;
  }
  while(i2 < 100);
  return 0;
}
```

file_io.cc

```
program file_io
  integer :: i, n
  real,allocatable :: T(:), rho(:)
  n = num_lines("data.txt")
  allocate(T(n), rho(n))
  open (unit=5, file="data.txt", &
        action="read")
  do i = 1, n
    read(5, *) T(i), rho(i)
  end do
end program file_io
integer function num_lines(s)
  character(len=*) :: s
  integer :: io=0
  num_lines=0
  open(unit=5,file=s,action="read")
  do while (1 .eq. 1)
    read(5, *, iostat=io)
    if (io < 0) exit
    num_lines = num_lines + 1
  end do
  close(unit=5)
end function num_lines
```

file_io.f90