Structures

Victor Eijkhout and Charlie Dey

spring 2017

Structures

Bundling information

Sometimes a number of variables belong logically together. For instance two doubles can be the x, y components of a vector.

This can be captured in the *struct* construct.

```
struct vector { double x; double y; } ;
```

(This can go in the main program or before it.)

Initialize:

```
struct vector { double x=0.; double y=0.; } ;
```

Using structures

Once you have defined a structure, you can make variables of that type. Setting and initializing them takes a new syntax:

```
struct vector p1,p2;

  p1.x = 1.; p1.y = 2.;
  p2 = {3.,4.};

  p2 = p1;
```

Functions on structures

You can pass a structure to a function:

```
double distance( struct vector p1,struct vector p2 ) {
double d1 = p1.x-p2.x, d2 = p1.y-p2.y;
return sqrt( d1*d1 + d2*d2 );
}
```

Returning structures

You can return a structure from a function:

```
struct vector vector_add
   ( struct vector p1,struct vector p2 ) {
struct vector p_add = {p1.x+p2.x,p1.y+p2.y};
return p_add;
};
```

(Something weird here with scopes: the explanation is that the returned value is copied.)

Exercise 1

Write a function `inner_product` that takes two `vector` structures and computes the inner product.

Exercise 2

Write a 2×2 matrix class (that is, a structure storing 4 real numbers), and write a function `multiply` that multiplies a matrix times a vector.

Project Exercise 3

Rewrite the exercise that found a predetermined number of primes, putting the `number_of_primes_found` and `last_number_tested` variables in a structure. Your main program should now look like:

```
struct primesequence sequence;
while (sequence.number_of_primes_found<nprimes) {
  int number = nextprime(sequence);
  cout << "Number " << number << " is prime" << endl;
}
```