

Structures

Victor Eijkhout, Susan Lindsey

COE 322 Fall 2020

Structures

1 Bundling information

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

This can be captured in the `struct` construct.

```
struct Point {  
    double x; double y; int label;  
};
```

(This is a type definition; it can go in the main program or before it.)

The elements of a structure are usually called members.

2 How to use structures

1. Define the structure: what members are in it.
2. Declare some structure variables;
3. Use these variables.

```
// definition of the struct
struct StructName { int num; double val; }
int main() {
    // declaration of struct variables
    StructName mystruct1,mystruct2;
    .... code that uses your structures ....
}
```

3 Using structures

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

Code:

```
int main() {  
  
    struct Point p1,p2;  
  
    p1.x = 1.; p1.y = 2.; p1.label =  
        5;  
    p2 = {3.,4.,5};  
  
    p2 = p1;  
    cout << "p2: "  
        << p2.x << ", " << p2.y  
        << endl;
```

Output

[struct] point:

```
./point  
p2: 1,2
```

Period syntax: compare to possessive 'apostrophe-s' in English.

Exercise 1

Define a struct `GridPoint` that models a grid point, meaning a point with integer coordinates. The Manhattan distance is defined as the sum of (absolute values of) the components. Read the values of the two components, and if the x component is more than the y , print 'this location is cross-town', otherwise print 'this location is uptown'.

You can base this off the file `gridpoint`

Review quiz 1

True or false?

- All members of a struct have to be of the same type.

```
/poll "struct members all the same type" "T" "F"
```

- Writing

```
struct numbered { int n; double x; };
```

creates a structure with an integer and a double as members.

```
/poll "'struct numbered { int n; double x; };' creates struct with int and double" "T" "F"
```

- All members of a struct have to be of different types.

```
/poll "All struct members have to be different types" "T" "F"
```

- With the above definition and `struct numbered xn;`,

```
cout << xn << endl;
```

Is this correct C++?

```
/poll "'cout << xn << endl;' legal?" "T" "F"
```

- With the same definitions, is this correct C++?

```
xn.x = xn.n+1;
```

```
/poll "'xn.x = xn.n+1;' legal?" "T" "F"
```

4 Struct initialization

You assign a whole struct, or set defaults in the definition.

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

```
// initialization when you create the variable:
```

```
struct Point_a x_a = {1.5,2.6};
```


5 Functions on structures

You can pass a structure to a function:

Code:

```
double distance
( struct point p1,
  struct point p2 )
{
    double
        d1 = p1.x-p2.x, d2 = p1.y-p2.y
        ;
    return sqrt( d1*d1 + d2*d2 );
}
/* ... */
cout << "dx=" << dx
      << ", dy=" << dy << endl;
struct point p2 = { p1.x+dx,p1.y
                  +dy };
cout << "Distance: "
      << distance(p1,p2) << endl;
```

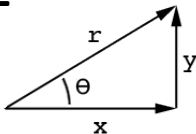
Output

[struct] pointfun:

dx=5, dy=12

Distance: 13

Exercise 2



Write a point structure, and a function that, given such a point, returns the angle with the x -axis. (Hint: the `atan` function is in `cmath`)

Code:

```
struct point a = {1.,1.};
double
    alpha = angle(a),
    pifrac = (4.*atan(1.0)) / alpha;
cout << "Angle of ("
    << a.x << ", " << a.y
    << ") is:\n " << angle(a)
    << ", or pi/" << pifrac
    << endl;
```

Output

[struct] pointangle:

```
Angle of (1,1) is:
    0.785398, or pi/4
Angle of (0.866025,0.5) is:
    0.523599, or pi/6
```

Exercise 3

Write a `void` function that has a `struct Point` parameter, and exchanges its coordinates:

$$\begin{pmatrix} 2.5 \\ 3.5 \end{pmatrix} \rightarrow \begin{pmatrix} 3.5 \\ 2.5 \end{pmatrix}$$

Code:

```
struct point a = {3.,2.};  
cout << "Flip of ("  
    << a.x << ", " << a.y << ")";  
flip(a);  
cout << " is ("  
    << a.x << ", " << a.y  
    << ")" << endl;
```

Output

[struct] pointflip:

Flip of (3,2) is (2,3)

6 Returning structures

You can return a structure from a function:

Code:

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

/* ... */
v3 = point_add(p1,p2);
cout << "Added: " <<
    v3.x << ", " << v3.y << endl;
```

Output

[struct] pointadd:

Added: 5,6

(In case you're wondering about scopes and lifetimes here: the explanation is that the returned value is copied.)

Exercise 4

Write a function $y = f(x, a)$ that takes a `struct Point` and `double` parameter as input, and returns a point that is the input multiplied by the scalar.

$$\begin{pmatrix} 2.5 \\ 3.5 \end{pmatrix}, 3 \rightarrow \begin{pmatrix} 7.5 \\ 10.5 \end{pmatrix}$$

7 Denotations

You can use initializer lists as struct *denotations* (with the *distance* function previous defined):

Code:

```
struct point p1{1.,2.}, p2
    {6.,14.};
cout << "Distance: "
    << distance( p1,p2 )
    << endl;
cout << "Distance: "
    << distance( {1.,2.},
    {6.,14.} )
    << endl;
```

Output

[struct] pointdenote:

```
Distance: 13
Distance: 13
```

Exercise 5

Write a function *inner_product* that takes two *Point* structures and computes the inner product. Test your code on some points where you know the answer.

Exercise 6

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.

Can you make a matrix structure that is based on the vector structure (essentially the same as a *Point* struct), for instance using vectors to store the matrix rows, and then using the inner product method to multiply matrices?

Project Exercise 7

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:

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

Hint: the variable `last_number_tested` does not appear in the main program. Where does it get updated? Also, there is no update of `number_of_primes_found` in the main program. Where do you think it would happen?

Turn it in!

- If you have compiled your program, do:

```
coe_struct yourprogram.cc
```

where 'yourprogram.cc' stands for the name of your source file.

- Is it reporting that your program is correct? If so, do:

```
coe_struct -s yourprogram.cc
```

where the -s flag stands for 'submit'.

- If you don't manage to get your code working correctly, you can submit as incomplete with

```
coe_struct -i yourprogram.cc
```