Structures

Kevin Schmidt, Susan Lindsey, Charlie Dey

Spring 2019

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; int label; } ;
```

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

The elements of a structure are usually called *members*.

How to use structures

1. Declare what is in your structure;
2. Make some structures;
3. Use them.

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

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 vector v1,v2;

  v1.x = 1.; v1.y = 2.; v1.label = 5;
  v2 = {3.,4.,5};

  v2 = v1;
  cout << "v2: " << v2.x << "," << v2.y << endl;
```

Output
[struct] point:

```
./point
v2: 1,2
```

Period syntax: 'apostrophe-s'.

Review quiz 1

True or false?

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

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

  creates an object with an integer and a double as members.
- With the above definition and `struct numbered xn;`,

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

  is correct C++.
- Same,

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

Struct initialization

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

```
struct vector_a { double x; double y; } ;
// needs compiler option: -std=c++11
struct vector_b { double x=0; double y=0; } ;

int main() {

  // initialization when you create the variable:
  struct vector_a x_a = {1.5,2.6};
  // initialization done in the structure definition:
  struct vector_b x_b;
  // ILLEGAL:
  // x_b = {3.7, 4.8};
  x_b.x = 3.7; x_b.y = 4.8;
```

Functions on structures

You can pass a structure to a function:

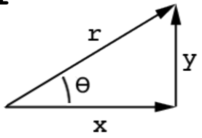
Code:

```
double distance
  ( struct vector v1,
    struct vector v2 )
{
  double
    d1 = v1.x-v2.x, d2 = v1.y-v2.y;
  return sqrt( d1*d1 + d2*d2 );
}
  /* ... */
  struct vector v1 = { 1.,1. };
  cout << "Displacement x,y?";
  double dx,dy; cin >> dx >> dy; cout << endl;
  cout << "dx=" << dx << ", dy=" << dy << endl;
  struct vector v2 = { v1.x+dx,v1.y+dy };
  cout << "Distance: " << distance(v1,v2) << endl;
```

Output
[struct] pointfun:

```
Displacement x,y?
dx=5, dy=12
Distance: 13
```

Exercise 1



Write a function that, given a vector as defined above, returns the angle with the *x*-axis. (Hint: the `atan` function is in `cmath`)

Code:

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

Output
[struct] pointangle:

```
Angle of (1,1) is 0.785398, or p
Angle of (0.866025,0.5) is 0.52
```

Exercise 2

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

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

Code:

```
struct vector 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)
```

Returning structures

You can return a structure from a function:

Code:

```
struct vector vector_add
      ( struct vector v1,
        struct vector v2 ) {
   struct vector p_add =
     {v1.x+v2.x,v1.y+v2.y};
   return p_add;
};
  /* ... */
  v3 = vector_add(v1,v2);
  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 3

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

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

Denotations

You can use initializer lists as struct *denotations*:

Code:

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

Output
[struct] pointdenote:

```
Displacement x,y?
dx=5, dy=12
Distance: 13
```

Exercise 4

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

Exercise 5

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, for instance using vectors to store the matrix rows, and then using the inner product method to multiply matrices?

Project Exercise 6

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:

  `sdsteststruct yourprogram.cc`

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

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

  `sdsteststruct -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

  `sdsteststruct -i yourprogram.cc`