8 Structures

Kevin Schmidt, Susan Lindsey, Charlie Dey

Bundle Information in a `struct`

- Sometimes a number of variables belong logically together. For instance two doubles can be the `x,y` components of a point.
- The `struct` construct bundles one or more data types together.
- The elements of a structure are usually called members.

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

Structure syntax

Definition:

```
struct Struct_name {
   datatype struct_var1
   datatype struct_var2};
```

Declaration:

```
Struct_name mystruct;
```

Initialize:

```
mystruct.struct_var1 = somevalue;
mystruct.struct_var2 = anothervalue;
```

use dot "." syntax to access struct members

Using Structures

- Define the structure itself at the top of the program, before functions.

```
struct StructName {int num; double val;}
```

- Declare and initialize variables of type `StructName` in your main program, as usual:

```
int main() {
   StructName mystruct1,mystruct2;
   mystruct1.val=2.5; mystruct.num=10;
   myfunc(mystruct);
}
```

Initializing Structures

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

void printpoint(Point p) {
  cout << "(" << p.x << "," \
       << p.y << ")" << endl;}

int main() {
    Point p1,p2;

    p1.x=3; p1.y=5;
    printpoint(p1);

    p2={18,10};
    printpoint(p2);
}
```

```
$ ./a.out
(3,5)
(18,10)
```

Review Quiz

True or false?

- The members of a `struct` must be of the same type.
- The following definition:

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

  creates an object with an integer and a double as members.
- Given this declaration:

  ```
  struct numbered xn;
  ```

  are these statements correct?
  - `cout << xn << endl;`
  - `xn.x = xn.n+1;`
  - `xn = {5,3.5};`

Passing Structures to Functions

Just like any other data type, you can *pass* structures to functions.

```
double distance(Vector v1, Vector v2 ){
   double d1 = v1.x-v2.x;
   double d2 = v1.y-v2.y;
   return sqrt( d1*d1 + d2*d2 );
}
main {
   Vector vectorA, vectorB;
   vectordistance = distance (vectorA, vectorB);
}
```

Returning Structures from Functions

And, just like any other data type, you can *return* structures from functions.

```
// structure definition
struct vector { double x; double y; } ;
// function definition
struct vector vector_add (vector v1, vector v2 ) {
    vector sum = {v1.x+v2.x,v1.y+v2.y};
    return sum;
};
// in main...
    vector v1 = {2., 2.}; //declare & initialize v1
    vector v2 = {3., 6.};
    vector vectorsum = vector_add (v1, v2);
```

Exercise 1

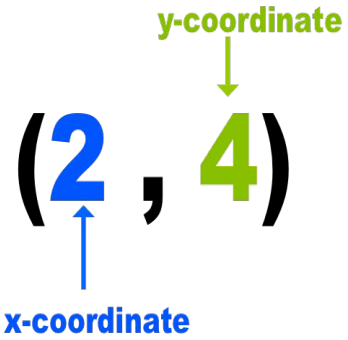
Write a void function, `flip`, that has a struct `Point` parameter, and exchanges its `x` and `y` member coordinates:

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

Struct allows you to create your own datatypes with int, float, double, string, and/or bool

Current Code:

```
int x=2;
int y=4;
```



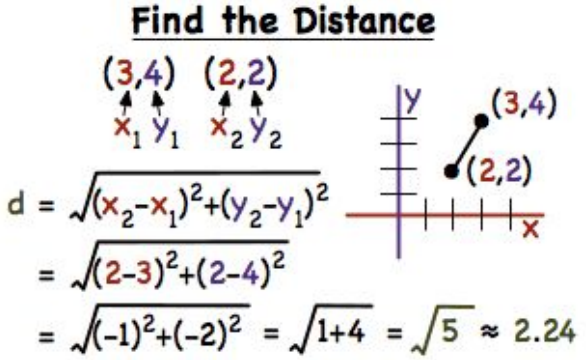
In Structure Code:

```
struct coord {int x;
int y; };
coord point_1;
point_1.x = 2;
point_1.y =4;

coord point_2;
point_2.x = 0;
point_2.y = 4;
```

Example

Distance between two points




```
#include<iostream>
#include<math.h>
using std::cout;
using std::endl;
int main() {
float x1 = 3.0;
float y1 = 4.0;
float x2 = 2.0;
float y2 = 2.0;
float dist =
sqrt(pow(x2-x1,2)+pow(y2-y1
,2));
cout << dist << endl;
}
```

Example with Struct

```
...
struct coords {
   float x;
   float y;
};
int main() {
   coords c1,c2;
   c1.x = 3.;
   c1.y = 4.;
   c2.x = 2.;
   c2.y = 2.;
   float dist = sqrt(pow(c2.x-c1.x,2)+pow(c2.y-c1.y,2));
   cout << dist << endl;
}
```

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 another `matrix`.

Remember, exercise 2 from the HW04.

Exercise 3

Create a new structure, `Point3d`, that contains `x`, `y`, and `z` real coordinates, then write a function, `distance3d`, that calculates the distance between those two `Point3d`'s using the distance formula in 3 dimensions:

$$AB=\sqrt{(x2-x1)^2+(y2-y1)^2+(z2-z1)^2};$$

```
float distance = distance3d(P1,P2)
```