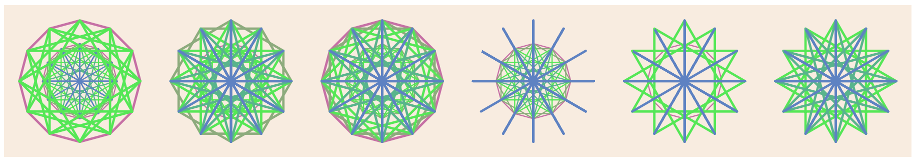
Exercises

Leon Tabak

26 May 2022

This work is licensed under CC BY 4.0. To view a copy of this license, visit http://creativecommons.org/licenses/by/4.0/.



Today's problem.

Write a program that stores and then prints the locations of several cities in China and the United States.

Specify each location with a **latitude** and a **longitude**.

Latitude is the distance north or south of the equator.

- A latitude may have any value between −90.0 (the South Pole) and +90.0 (the North Pole).
- Latitudes north of the equator are positive.
- Latitudes south of the equator are negative.

Longitude is the distance (measured in degrees) east or west of the prime meridian. The prime meridian is a line that runs from the North Pole to the South Pole, passing through Greenwich, England..

- A longitude may have any value between −180.0 and +180.0.

  (That's not quite right. There is no longitude whose value is −180 degrees.

  A point that is 180 degrees west of the prime meridian is also 180 degrees east of the prime meridian. Geographers specify the longitude of a such a point with the value +180 degrees. They never write −180 degrees.)

- Longitudes east of the prime meridian are positive.

- Longitudes west of the prime meridian are negative.

Programming skills

In this exercise, you will practice using the following features of the C programming language:

- one and two dimensional arrays
- definitions and calls to functions
- **for** loops
- use of the printf() function and format strings to produce formatted output
- strings
- definition of new data structures with **struct**
- **typedef** to create aliases—that is, create alternative names for data types

Before you begin programming...

Search on the Internet for the latitude and longitude of several cities in China and the United States.

Version 0 of program

```
#include <stdio.h>
#include <stdlib.h>

struct geographicLocation {
  double latitude;
  double longitude;
};
```

```c
typedef struct geographicLocation coordinates;

typedef struct geographicLocation *coordinatesPointer;

coordinates makeCoordinates( double latitude,
        double longitude ) {

    coordinates result;
    result.latitude = latitude;
    result.longitude = longitude;

    return result;
} // makeCoordinates( double, double )

void printCoordinates( char* city, coordinates c ) {
    printf( "%20s (latitude = %8.2f, longitude = %8.2f)\n\n",
        city, c.latitude, c.longitude );
} // printCoordinates( char*, coordinates )

int main( int argc, char** argv ) {

    char americanCities[4][20] = {
        "Boston",
        "New York",
        "Atlanta",
        "Miami" };

    char chineseCities[4][20] = {
        "Jilin City",
        "Beijing",
        "Shanghai",
        "Hong Kong" };

    double americanLatitudes[] = {
        42.36,
        40.71,
        33.75,
        25.76 };

    double americanLongitudes[] = {
        -71.06,
        -74.01,
        -84.39,
        -80.19 };
```

```
 double chineseLatitudes[] = {
     43.84,
     39.91,
     31.23,
     22.32 };

double chineseLongitudes[] = {
     126.55,
     116.41,
     121.47,
     114.17 };

 printf( "\nAmerican cities\n" );
 for( int i = 0; i < 4; i++ ) {
     printf( "\t%20s (%8.2f, %8.2f)\n",
         americanCities[i],
         americanLatitudes[i],
         americanLongitudes[i] );
 } // for

 printf( "\nChinese cities\n" );
 for( int i = 0; i < 4; i++ ) {
     printf( "\t%20s (%8.2f, %8.2f)\n",
         chineseCities[i],
         chineseLatitudes[i],
         chineseLongitudes[i] );
 } // for


 /*
 // American cities
 coordinates boston;
 boston.latitude = 42.36;
 boston.longitude = -71.06;

 coordinates newYork;
 newYork.latitude = 40.71;
 newYork.longitude = -74.01;


 coordinates atlanta;
 atlanta.latitude = 33.75;
 atlanta.longitude = -84.39;

 coordinates miami;
 miami.latitude = 25.76;
miami.longitude = -80.19;

// Chinese cities
coordinates jilinCity;
jilinCity.latitude = 43.84;
jilinCity.longitude = 126.55;

coordinates beijing;
beijing.latitude = 39.91;
beijing.longitude = 116.41;

coordinates shanghai;
shanghai.latitude = 31.23;
shanghai.longitude = 121.47;

coordinates hongKong;
hongKong.latitude = 22.32;
hongKong.longitude = 114.17;
*/

printf( "\n" );

// American cities
coordinates boston = makeCoordinates( 42.36, -71.06 );
printCoordinates( "Boston", boston );

coordinates newYork = makeCoordinates( 40.71, -74.01 );
printCoordinates( "New York", newYork );

coordinates atlanta = makeCoordinates (33.75, -84.39 );
printCoordinates( "Atlanta", atlanta );

coordinates miami = makeCoordinates( 25.76, -80.19 );
printCoordinates( "Miami", miami );

// Chinese cities
coordinates jilinCity = makeCoordinates( 43.84, 126.55 );
printCoordinates( "Jilin City", jilinCity );

coordinates beijing = makeCoordinates( 39.91, 116.41 );
printCoordinates( "Beijing", beijing );

coordinates shanghai = makeCoordinates( 31.23, 121.47 );
printCoordinates( "Shanghai", shanghai );

coordinates hongKong = makeCoordinates( 22.32, 114.17 );
```

```
    printCoordinates( "Hong Kong", hongKong );

} // main( int, char** )
```

Version 1 of program

```
#include <stdio.h>
#include <stdlib.h>

struct geographicLocation {
  double latitude;
  double longitude;
};

typedef struct geographicLocation coordinates;

typedef struct geographicLocation *coordinatesPointer;

coordinatesPointer makeCoordinates( double latitude,
        double longitude ) {

    coordinatesPointer cp =
        (coordinatesPointer) calloc( 1, sizeof(coordinates) );

    cp->latitude = latitude;
    cp->longitude = longitude;

    return cp;
} // makeCoordinates( double, double )

void printCoordinates( char* city, coordinatesPointer cp ) {
    printf( "%20s (latitude = %8.2f, longitude = %8.2f)\n\n",
        city, cp->latitude, cp->longitude );
} // printCoordinates( char*, coordinatesPointer )

int main( int argc, char** argv ) {

    printf( "\n" );

    // American cities
    coordinatesPointer bostonPointer =
        makeCoordinates( 42.36, -71.06 );
    printCoordinates( "Boston", bostonPointer );

    coordinatesPointer newYorkPointer =
        makeCoordinates( 40.71, -74.01 );
    printCoordinates( "New York", newYorkPointer );

    coordinatesPointer atlantaPointer =
        makeCoordinates (33.75, -84.39 );
    printCoordinates( "Atlanta", atlantaPointer );

    coordinatesPointer miamiPointer =
        makeCoordinates( 25.76, -80.19 );
    printCoordinates( "Miami", miamiPointer );

    // Chinese cities
    coordinatesPointer jilinCityPointer =
        makeCoordinates( 43.84, 126.55 );
    printCoordinates( "Jilin City", jilinCityPointer );

    coordinatesPointer beijingPointer =
        makeCoordinates( 39.91, 116.41 );
    printCoordinates( "Beijing", beijingPointer );

    coordinatesPointer shanghaiPointer =
        makeCoordinates( 31.23, 121.47 );
    printCoordinates( "Shanghai", shanghaiPointer );

    coordinatesPointer hongKongPointer =
        makeCoordinates( 22.32, 114.17 );
    printCoordinates( "Hong Kong", hongKongPointer );

    free( bostonPointer );
    free( newYorkPointer );
    free( atlantaPointer );
    free( miamiPointer );

    free( jilinCityPointer );
    free( beijingPointer );
    free( shanghaiPointer );
    free( hongKongPointer );
} // main( int, char** )
```

Questions

1. Examine this code.

```
        printf( "\nChinese cities\n" );
        for( int i = 0; i < 4; i++ ) {
            printf( "\t%20s (%8.2f, %8.2f)\n",
            chineseCities[i],
            chineseLatitudes[i],
            chineseLongitudes[i] );
    } // for
```

Match each symbol on the left with its description on the right.

%8.2f	floating point value with 8 characters
%20s	new line character
\n	string with 20 characters
\t	tab character

2. Which is the correct way of initializing the fields in this case?

```
// Define a data structure.
struct location {
  double latitude;
  double longitude:
};

// Create a variable of that type.
struct location jilinCity;
// Jilin City 43.84N, 126.55 E
```

   (a) Use a period between name of variable and field?

   ```
   // Initialize the fields of the variable.
   jilinCity.latitude = 43.84;
   jilinCity.longitude = 126.55;
   ```

   (b) Or use an arrow?

   ```
   // Initialize the fields of the variable.
   jilinCity->latitude = 43.84;
   jilinCity->longitude = 126.55;
   ```

3. Which is the correct way of initializing the fields in this case?

```
// Define a data structure.
struct location {
  double latitude;
  double longitude:
};

// Create a variable whose type
// is a pointer to that kind of data structure.
struct location *jcp =
      (struct location*) calloc( 1, sizeof(location) );
```

(a) Use a period between name of variable and field?

```
// Initialize the fields of the variable.
jcp.latitude = 43.84;
jcp.longitude = 126.55;
```

(b) Or use an arrow?

```
// Initialize the fields of the variable.
jcp->latitude = 43.84;
jcp->longitude = 126.55;
```

4. In which kind of data structure does every element have the same type?
   (a) array
   (b) **struct**
5. In which kind of data structure does different elements have different types?
   (a) array
   (b) **struct**
6. With which kind of data structure does a programmer identifying elements with a name?
   (a) array
   (b) **struct**
7. With which kind of data structure does a programmer identifying elements with a numerical index?
   (a) array
   (b) **struct**