CS 101: Computer Programming and Utilization

16-Structures

Instructor: Sridhar Iyer
IIT Bombay
Activity

Write a program to manage your contacts info:

• You need to store the following information about each contact – Name, Phone Number, Email.
• You need provide functions to – Input, Lookup, Update and Delete information about contacts.

Think: How will you store the information?

Pair: Write the C++ declarations. Write pseudo-code for the functions.

Share: Class discussion of implementations.
Suppose we use 3 arrays

char name[100][10]; // 100 names of 10 char each.
long number[100]; // can int hold 10 digit phone#?
string email[100]; // how long can the string be?

Entry at index 'i' in each array will have information about the i-th contact.

We need to be careful while implementing functions, to ensure that we are accessing the correct entry in each array. Minor bug can cause major mix-ups.

So we want a feature like a “spreadsheet row”

<table>
<thead>
<tr>
<th>Name</th>
<th>Number</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>XYZ</td>
<td>10101</td>
<td><a href="mailto:xyz@abc.com">xyz@abc.com</a></td>
</tr>
</tbody>
</table>
We often need to access and process different pieces of information related to the same entity:

- Name, phone number, email of a contact;
- Roll, name, batch, marks of a student;

`struct` is useful in representing entities which have many attributes of distinct types.

```c
struct contactinfo {
    char name[10];
    long number;
    string email;
}

struct studentinfo {
    int roll;
    char name[30];
    int batch;
    float marks;
}
```
struct – variables and arrays

The struct definition is only the declaration of a new data type and its composition. We have to then define variables of that data type.

```c
struct contactinfo friend; // Declares a variable 'friend' of type contactinfo
```

- Attributes are accessed using . (dot) operator:
  - `friend.name = “xyz”` // Sets the name field to “xyz”
  - `cout << friend.number;`

```c
struct studentinfo students[450]; // Declares an array of type studentinfo
```

- `students[32].marks = 80.5`
Visibility of struct types

```c
void f()
{
  struct s{...};
  s s1, s2;
}
int main()
{
  s s3; // Error.
}
```

```c
struct s{...};
void f()
{
  s s1, s2; // Allowed
}
int main()
{
  s s3; // Allowed
}
```

If a structure type name is needed in several blocks, define it outside the blocks and above them in the file. Rules for visibility of struct are the same that of variables.
struct – more examples

struct Book{
    char title[50]; char author[50]; double price;
};

Book xyz = {“CS101”, “Phatak”, 100.0};

struct Point {double x, y;};
struct Circle {Point center; double radius};

Circle c ={{10,20}, 5};
circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copied!

circle c ={{10,20}, 5};
c1 = c2; // all members copies!
Returning structures

Circle hybrid(Circle c1, Circle c2){
    Circle c;
    c.center = c1.center;
    c.radius = c2.radius;
    return c; // New circle is created and returned
}

int main(){
    Circle c1={..}, c2 = {..}, c3;
    c3 = hybrid(c1, c2);
}
structures and functions

struct Circle{...};
bool intersect(Circle c1, Circle c2){
    return (pow(c1.radius + c2.radius, 2) 
        >= pow(c1.center.x – c2.center.x, 2) + 
        pow(c1.center.y – c2.center.y, 2));
}
int main(){
    Circle d1={{10,20},5}, d2 = {{20,20},5};
    cout << intersect(d1, d2) << endl;
}
Revisit your contacts database program

Revisit the contacts database program that you wrote at the beginning of today's class. You have now seen the struct declaration, and functions read(), write().

Think: Write the pseudo-code for functions - Lookup, Update, and Remove – using the struct.

Pair: Discuss your pseudo-code with your neighbour and see if you have missed any point(s). Together, write the C++ code for the functions.

Share: Compare with demo16-struct.cpp.