11.9: Pointers to Structures

- Given the following Structure:

  ```
  struct Student {
      string name;      // Student’s name
      int idNum;        // Student ID number
      int creditHours;  // Credit hours enrolled
      float gpa;        // Current GPA
  }
  ```

- We can define a pointer to a structure

  ```
  Student s1 = {"Jane Doe", 12345, 15, 3.3};
  Student *studentPtr;
  studentPtr = &s1;
  ```

- Now `studentPtr` points to the `s1` structure.

Pointers to Structures

- How to access a member through the pointer?

  ```
  Student s1 = {"Jane Doe", 12345, 15, 3.3};
  Student *studentPtr;
  studentPtr = &s1;
  cout << *studentPtr.name << end; // ERROR
  ```

  - dot operator has higher precedence than the dereferencing operator, so:
    ```
    *studentPtr.name is equivalent to *(studentPtr.name)
    ```
  - You must dereference the pointer first:

    ```
    cout << (*studentPtr).name << end; // WORKS
    ```

  - Due to the awkwardness of the pointer notation, C provides an operator for dereferencing structure pointers:

    ```
    studentPtr->name is equivalent to (*studentPtr).name
    ```

  - The **structure pointer operator** is the hyphen (-) followed by the greater than (>) like an arrow.

  - In summary:

    ```
    s1.name // a member of structure s1
    sptr->name // a member of the structure sptr points to
    ```
Structure Pointer: example

- Function to input a student, using a ptr to struct

```c++
void inputStudent(Student *s) {
    cout << "Enter Student name: ";
    getline(cin, s->name);
    cout << "Enter studentID: ";
    cin >> s->idNum;
    cout << "Enter credit hours: ";
    cin >> s->creditHours;
    cout << "Enter GPA: ";
    cin >> s->gpa;
}
```

- Call:

```c++
Student s1;
inputStudent(&s1);
cout << s1.name << endl;
...
```

Dynamically Allocating Structures

- Structures can be dynamically allocated with new:

```c++
Student *sPtr;
sPtr = new Student;
sPtr->name = "Jane Doe";
sPtr->idNum = 12345;
...
delete sPtr;
```

- Arrays of structures can also be dynamically allocated:

```c++
Student *sPtr;
sPtr = new Student[100];
sPtr[0].name = "John Deer";
...
delete [] sPtr;
```

in 13.3: Pointers to Objects

- We can define pointers to objects, just like pointers to structures

```c++
Time t1(12,20);
Time *timePtr;
timePtr = &t1;
```

- We can access public members of the object using the structure pointer operator (->)

```c++
timePtr->addMinute();
cout << timePtr->display() << endl;
```

Output:

```
l2:21
```

Dynamically Allocating Objects

- Objects can be dynamically allocated with new:

```c++
Time *tptr;
tptr = new Time(12,20);
...
delete tptr;
```

- Arrays of objects can also be dynamically allocated:

```c++
Time *tptr;
tptr = new Time[100];
tptr[0].addMinute();
...
delete [] tptr;
```

You can pass arguments to a constructor using this syntax.

It uses the default constructor to initialize the elements in the new array.

Initializer lists are not allowed.