Chapter 11

structs (C++ records)
In CS 1A we worked with a database program. We viewed the data in the database as being stored in records. Each record was a collection of fields. A record then was a complete set of information about a person, place or object and the fields defined the individual components of the record. C++ refers to records as *structs*.

We defined an array as a collection of information all of the same type (homogeneous). A struct then is a collection of information of different data types (heterogeneous). The fields of a struct are referred to as *members*.

```
struct StructName
{
    dataType memberName;
    .
    .
};

Example:

struct StudentRec
{
    string name;
    string idNum;
    float gpa;
};

StudentRec theStudent;
```

```
The individual members of the struct must be accessed by the name of the struct followed by the name of the member.

```cpp
theStudent.name = "Sally";
cin >> theStudent.idNum;
cout << theStudent.gpa;
```

Example:

```cpp
// This program demonstrates the use of a record (C++ struct)
#include <iostream.h>

struct PersonRec
{
    string lastName;
    string firstName;
    int age;
};

void main(void)
{
    PersonRec thePerson;

    cout << "Enter first name: ";
cin >> thePerson.firstName;
    cout << "Enter last name: ";
cin >> thePerson.lastName;
    cout << "Enter age: ";
cin >> thePerson.age;

    cout << "\nHello " << thePerson.firstName << " " << thePerson.lastName << ".  How are you?\n"
    cout << "Congratulations on reaching the age of " << thePerson.age << ".\n";
}
```
// This program demonstrates the use of a nested struct
struct GradeRec
{
    float percent;
    char grade;
};

struct StudentRec
{
    string lastName;
    string firstName;
    int age;
    GradeRec courseGrade;
};

void main(void)
{
    StudentRec student;

    cout << "Enter first name: ";
    cin >> student.firstName;
    cout << "Enter last name: ";
    cin >> student.lastName;
    cout << "Enter age: ";
    cin >> student.age;
    cout << "Enter overall percent: ";
    cin >> student.courseGrade.percent;
    if(student.courseGrade.percent >= 90)
    {
        student.courseGrade.grade = 'A';
    }
    else if(student.courseGrade.percent >= 75)
    {
        student.courseGrade.grade = 'B';
    }
    else
    {
        student.courseGrade.grade = 'F';
    }

    cout << '\n\nHello " << student.firstName << ' ' " << student.lastName
<< ". How are you?\n";
    cout << '\nCongratulations on reaching the age of " << student.age
<< ",\n";
    cout << "Your overall percent score is "
<< student.courseGrade.percent << " for a grade of "
<< student.courseGrade.grade;
}

OUTPUT:
Enter first name: Sally
Enter last name: Smart
Enter age: 19
Enter overall percent: 98

Hello Sally Smart. How are you?
Congratulations on reaching the age of 19.
Your overall percent score is 98 for a grade of A
// This program demonstrates the use of an array of structs
#include <iostream.h>

struct PersonRec
{
    string lastName;
    string firstName;
    int age;
};
typedef PersonRec PeopleArrayType[10]; // an array of 10 structs

void main(void)
{
    PeopleArrayType people;     // a variable of the array type

    for (int i = 0; i < 10; i++)
    {
        cout << "Enter first name: ";
        cin >> people[i].firstName;
        cout << "Enter last name: ";
        cin >> people[i].lastName;
        cout << "Enter age: ";
        cin >> people[i].age;
    }

    for (int i = 0; i < 10; i++)
    {
        cout << people[i].firstName << ' ' << people[i].lastName
             << setw(10) << people[i].age;
    }
}
#include <iostream.h>

struct PersonRec
{
    string lastName;
    string firstName;
    int age;
};

typedef PersonRec PeopleArrayType[10];  // an array of 10 structs

void LoadArray(PeopleArrayType peop);

void main(void)
{
    PeopleArrayType people;  // a variable of the array type

    LoadArray(people);

    // output the array
    for (int i = 0; i < 10; i++)
    {
        cout << people[i].firstName << ' ' << people[i].lastName
             << setw(10) << people[i].age;
    }
}

void LoadArray(PeopleArrayType peop)
{
    for (int i = 0; i < 10; i++)
    {
        cout << "Enter first name: ";
        cin >> peop[i].firstName;
        cout << "Enter last name: ";
        cin >> peop[i].lastName;
        cout << "Enter age: ";
        cin >> peop[i].age;
    }
}
**structs and Aggregate Operations**

- Aggregate I/O is not allowed. I/O must be performed on a member by member basis.

- Aggregate assignment is allowed. All data members (fields) are copied.

- Aggregate arithmetic is not allowed.

- Aggregate comparison is not allowed. Comparisons must be performed on a member by member basis.

- structs may be passed by value or by reference.

- A struct is a valid return type for a value returning function.
Palindrome Lab - struct

Write a C++ program to manage a user-defined string. The program will create a string from the input buffer, output the string and its length, and check the string to determine whether or not it is a palindrome. A palindrome is a string that reads the same forwards and backwards. Examples:

- radar
- racecar
- a man a plan a canal panama

```cpp
struct StringRec{
  int strLen;
  char theStr[256];
};

void AddChar(StringRec& str, char theCh);  // adds one character to the string
void OutputString(StringRec str);         // outputs the string and the length of the string
bool CheckString(StringRec str);          // returns true if string is a palindrome, false otherwise

void main(void)
{
  StringRec theString;
  char theChar;

  theString.strLen = 0;
  cout << "Enter a string: ";
  cin.get(theChar);
  while(theChar != '\n')
  {
    AddChar(theString, theChar);
    cin.get(theChar);
  }
  OutputString(theString);
  if( CheckString(theString) )
    cout << "\n\nThe string is a palindrome";
  else
    cout << "\n\nThe string is not a palindrome";
}
```