Structs and Classes

Arrays of Pointers

- What do you make of the following declaration?

```
char *cardSuits[4] = {"Clubs", "Diamonds",
                      "Hearts", "Spades"};
```

- What gets output in each of the following cases?

```
cout << cardSuits[1] << endl;

cout << *cardSuits[1] << endl;
```


Arrays of Pointers

- Let's look at an example where arrays of pointers are used
  - http://zeus.cs.pacificu.edu/shereen/cs250/Lectures/05card.cpp
- This example also serves as a reminder of how to generate random numbers

String Functions

- Let's take a look at the string functions on pp. 363

Review of structs

```
struct Person
{
  char nameStr[20];
  char ssNum[9];
  int age;
};
```

- What do each of the following declarations mean?

```
Person personStruct;
Person personArry[5];
Person *pPerson = &personStruct;
Person &personRef = personStruct;
```


References in C++

- `Person &personRef = personStruct;`
- A reference is like a constant pointer that is automatically dereferenced

```
int x = 0;
int &a = x;
cout << x << a << endl;
a++;
cout << x << a << endl;
```

Rules for References

- A reference must be initialized when it is created
- Once a reference is initialized to an object, it cannot be changed to refer to another object
- You cannot have NULL references

Accessing Structure Members

- How do we access structure members?
- How can I initialize the contents of personStruct using the object itself, the pointer, and the reference?

Classes

- The reserved word **`class`** is used to create the complex **`structure`**
- Classes differ from structures in that:
  - They don't just combine simple data types into one object
  - They also describe how that data can be manipulated

More on Objects

- Object-oriented programming hides the details of objects from objects of other types
- When an object needs information from another object or needs another object to perform a task, it sends a message to the object requesting what it needs
- As a result, object-oriented programs can be written more generically than structured programs
- Usually, making changes to the object-oriented programs is easier than changing structured programs

In Summary

- A **`class`** is like a **`struct`** but much more
- Whereas **`structs`** can contain simple data types, **`classes`** contain both *data types* and *functions* that manipulate the class data

A C++ Example

- Enough of theory!
- Let's have a look at a real example.
- We will create a **`class Person`** that will:
  - Store information about person
  - Store functions to manipulate this information

The Person Class

```
class Person
{
public:
   int age;

   int returnAge();
   int returnBirthYear();
};
int main()
{
   Person person;
   person.age = 28;
   cout << "person is: " << person.returnAge();
   cout << "person was born in: "
        << person.returnBirthYear();
   return 0;
}
```

Data Member

Member Function prototypes

The Person Class

```
int Person::returnAge()
{
  return age;
}

int Person::returnBirthYear()
{
  return 2003 - age;
}
```


Private & Public

- Class data members and member functions can be either private or public
- Private data members and member functions can only be accessed within that class
- Public data members and member functions can be accessed from outside of that class

Example Using Private & Public

```
class Person
{
private:
   int age;
public:
   void setAge(int);
   int returnAge();
   int returnBirthYear();
};
int main()
{
   Person person;
   person.setAge(28);
   cout << "person is: " << person.returnAge() << endl;
   cout << "person was born in: "
        << person.returnBirthYear();
   return 0;
}
```

- Because `age` is a private data member, we can't use `person.age = 28` here.
- Instead, we need to create a new function in the class to set the age.

Continued

```
void Person::setAge(int newAge)
{
  age = newAge;
}
```


Another Example

- Let's look at example 6.3 in the book

Summary

- Today I introduced
  - Review of structures
  - References
  - Classes
- We have covered:
  - P. 405 - 418 in Chapter 6