

## struct

*struct* is a structure that keeps objects of different types together as one object. For example if we wish to keep data on a student, we would need to know their name, age, gpa, address, phone number, etc. We would want to keep all this information together in one structure (sometimes called a record). Here is the C++ code to do so:

```
struct StudentInfo
{
    apstring name;           // Field names
    int age;
    double gpa;
    apstring address;
    apstring phone;
};

void Print(const StudentInfo & student);

int main ()
{
    StudentInfo s;

    s.name = "Bob";
    s.age = 16;
    s.gpa = 3.14;
    s.address = "750 Nautilus";
    s.phone = "454-3081";
    Print(s);

    return 0;
}

void Print(const StudentInfo & student)
// Pass as a constant reference parameter to save memory but
// prevent alterations
{
    cout << "Name: " << student.name << endl;
    cout << "Age: " << student.age << endl;
    // etc.
}
```

The struct is one record containing many objects. To access a particular object in a *struct* list the name of the *struct* followed by a dot followed by the field name (name of object in struct). The example above shows this syntax.

The struct now can be passed as one variable to a function. So instead of passing five separate variables, they can be group in a struct and passed as one variable.

A *struct* is the same as a *class*, except that all member functions and data are public. There are no private parts to a *struct*. Struct's can have member functions and constructors. Generally there are no member functions (otherwise it's probably better to use a class), but there usually is a constructor. This constructor is usually written "inline", that is the code for the constructor is written in the struct definition. For example:

```
struct Point
{
    Point() : myX(0), myY(0) {};           // default constructor
    Point(int x, int y) { myX = x; myY = y; };
    int myX;
    int myY;
};
```

The default constructor uses an initialization list to set the data values, the second constructor uses the normal executable way.