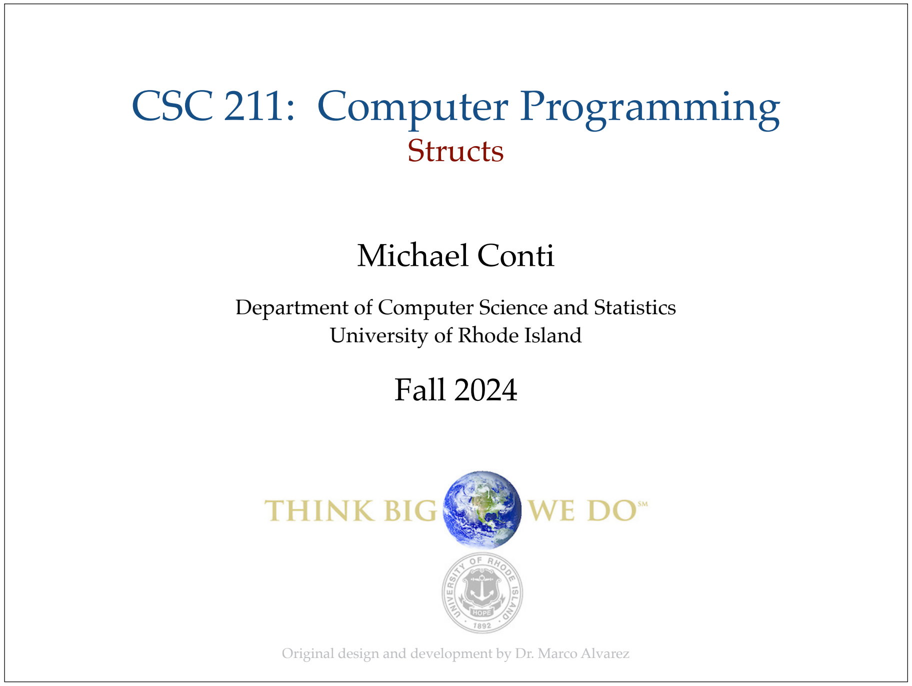


# CSC 211: Computer Programming

## Structs

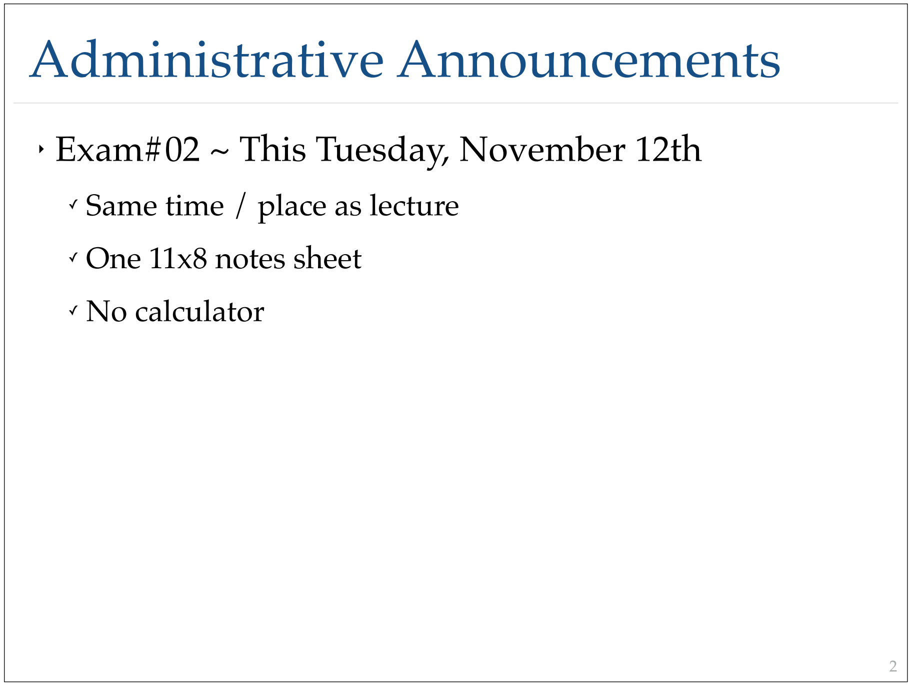
Michael Conti

Department of Computer Science and Statistics
University of Rhode Island

Fall 2024

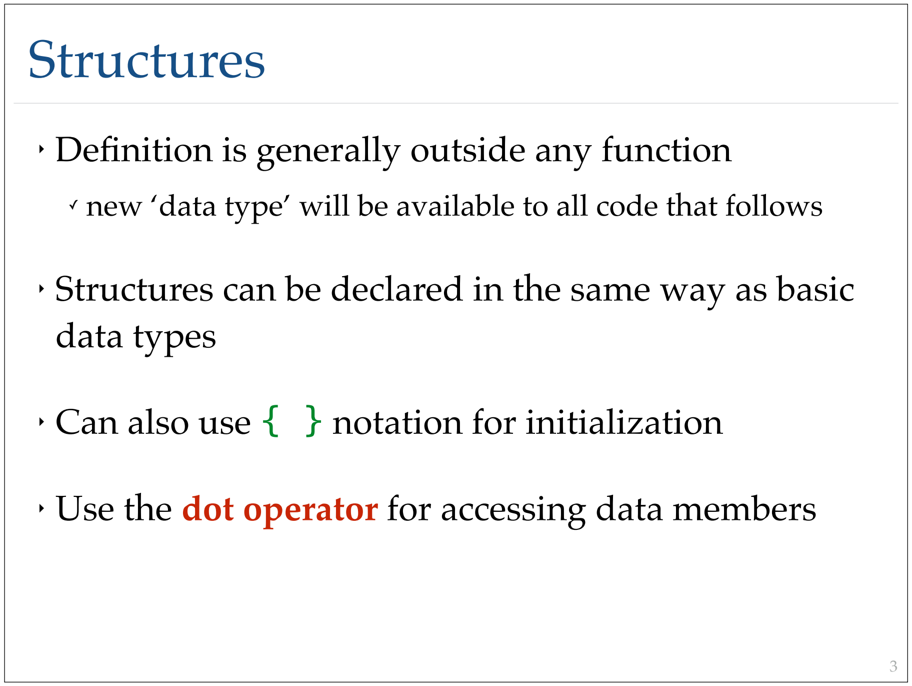
THINK BIG WE DO℠

Original design and development by Dr. Marco Alvarez



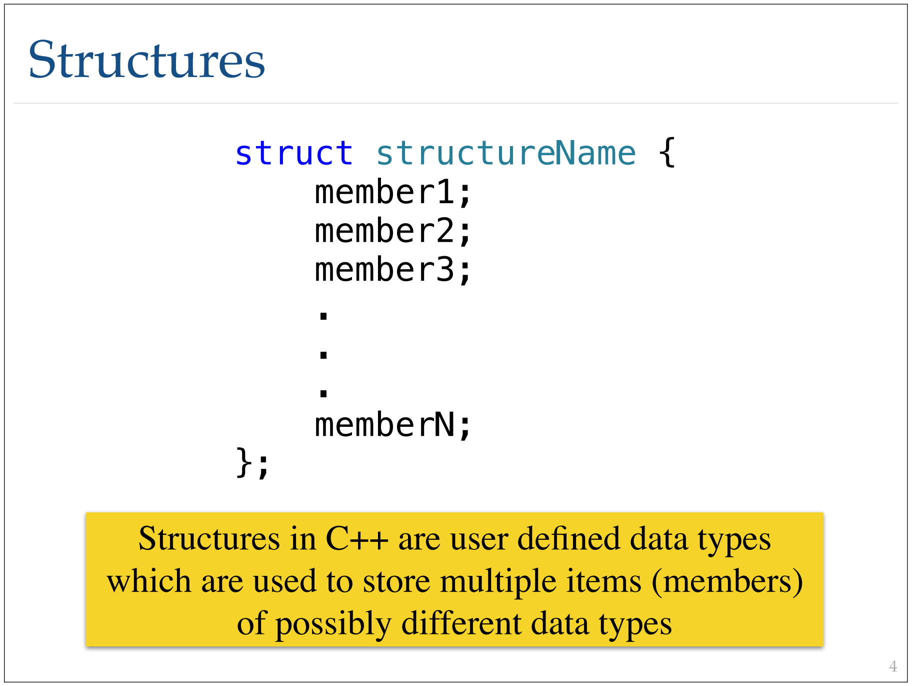
# Administrative Announcements

- Exam#02 ~ This Tuesday, November 12th
  - Same time / place as lecture
  - One 11x8 notes sheet
  - No calculator



# Structures

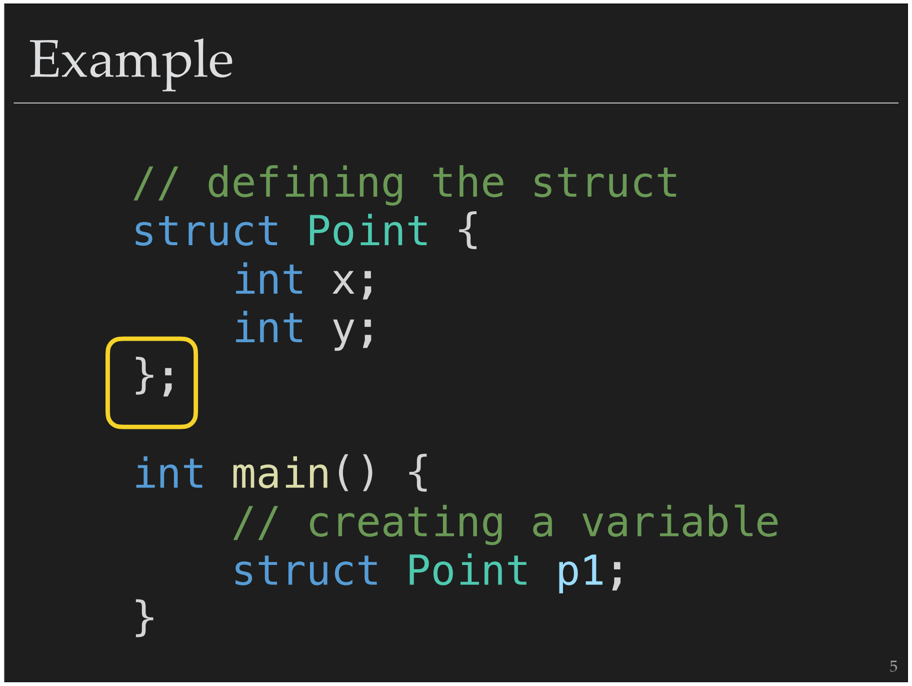
- Definition is generally outside any function
  - new ‘data type’ will be available to all code that follows
- Structures can be declared in the same way as basic data types
- Can also use { } notation for initialization
- Use the **dot operator** for accessing data members



# Structures

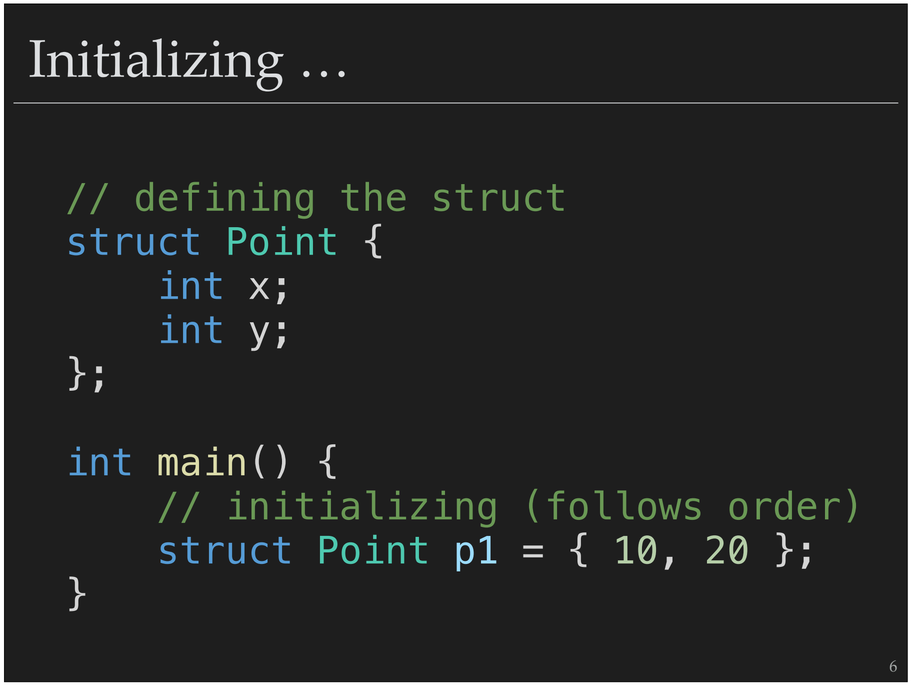
```
struct structureName {
    member1;
    member2;
    member3;
    .
    .
    .
    memberN;
};
```

Structures in C++ are user defined data types which are used to store multiple items (members) of possibly different data types



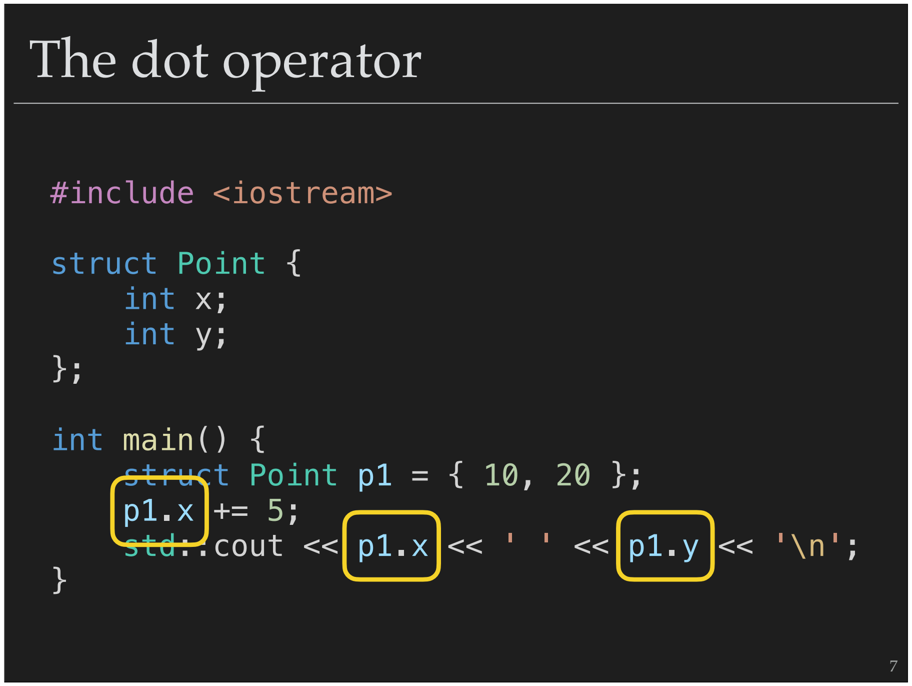
## Example

```
// defining the struct
struct Point {
    int x;
    int y;
};

int main() {
    // creating a variable
    struct Point p1;
}
```



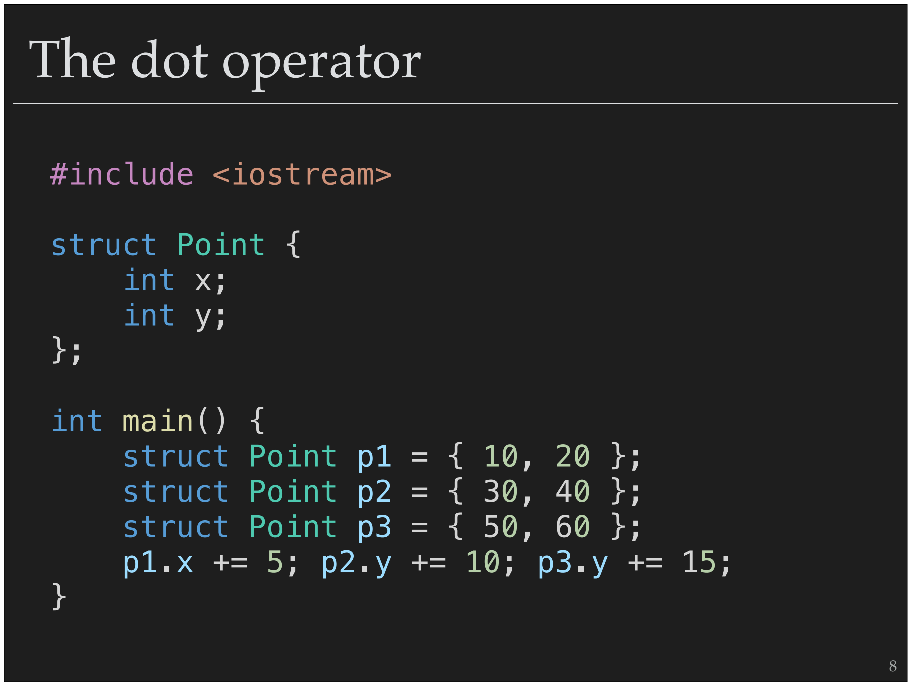
## Initializing …

```
// defining the struct
struct Point {
    int x;
    int y;
};

int main() {
    // initializing (follows order)
    struct Point p1 = { 10, 20 };
}
```



## The dot operator

```
#include <iostream>

struct Point {
    int x;
    int y;
};

int main() {
    struct Point p1 = { 10, 20 };
    p1.x += 5;
    std::cout << p1.x << ' ' << p1.y << '\n';
}
```

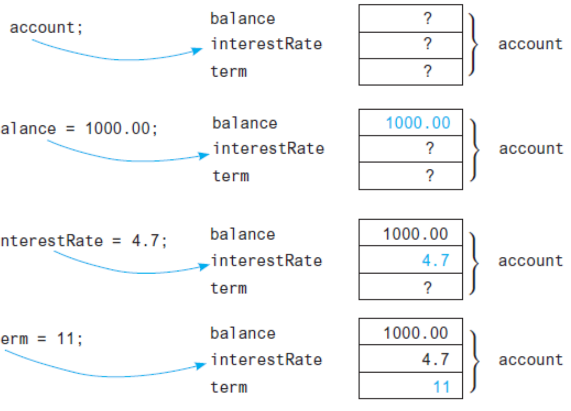


## The dot operator

```
#include <iostream>

struct Point {
    int x;
    int y;
};

int main() {
    struct Point p1 = { 10, 20 };
    struct Point p2 = { 30, 40 };
    struct Point p3 = { 50, 60 };
    p1.x += 5; p2.y += 10; p3.y += 15;
}
```

**DISPLAY 10.2** Member Values

```
struct CDAccount
{
    double balance;
    double interestRate;
    int term; //months until maturity
};
int main( )
{
    CDAccount account;
        ...

    account.balance = 1000.00;


    account.interestRate = 4.7;


    account.term = 11;
```

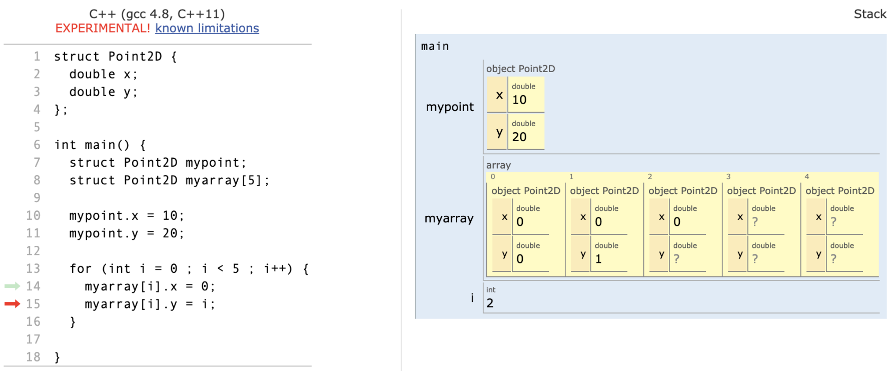


from: Problem Solving with C++, 10th Edition, Walter Savitch

# Array of structures

```
#include <iostream>

struct Point2D {
    double x;
    double y;
};

int main() {

    Point2D mypoint;
    Point2D myarray[5];

    mypoint.x = 10;
    mypoint.y = 20;

    for (int i = 0 ; i < 5 ; i ++) {
        myarray[i].x = 0;
        myarray[i].y = i;
    }
}
```

# pythontutor.com



```
struct Point2D {
  double x;
  double y;
};

int main() {
  struct Point2D mypoint;
  struct Point2D myarray[5];

  mypoint.x = 10;
  mypoint.y = 20;

  for (int i = 0 ; i < 5 ; i++) {
    myarray[i].x = 0;
    myarray[i].y = i;
  }

}
```

# Arrays and Structures

- When using arrays as structs member, the index goes at the end
  - student.grades[i]
- When using structs as arrays elements, the index goes after the struct name
  - students[i].finalGrade

# Functions

```
// defining the struct
struct Point {
     int x;
     int y;
};

void distance(Point P1,Point P2);
```

# Passing structures to functions

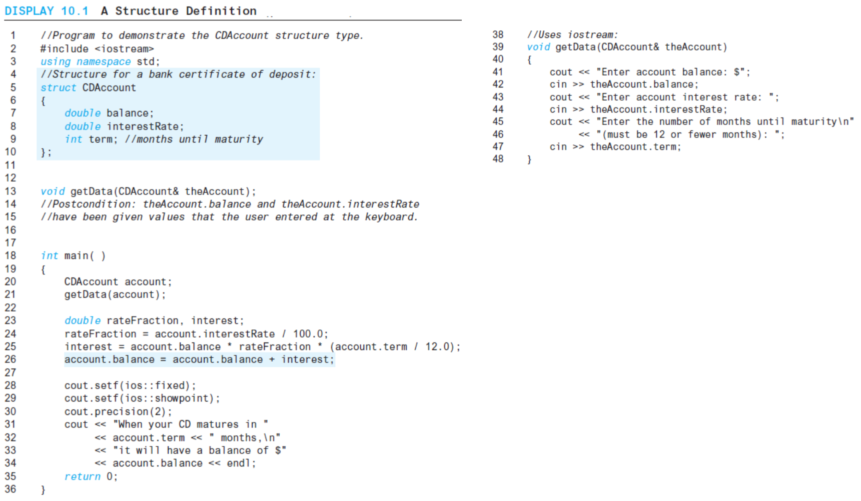
- A struct can be passed as a parameter either by value or by reference

```
void incrementPoint(Point &somePoint){
    somePoint.x+=1;
    somePoint.y+=1;
};
```

- A function can return a value of type struct

```
Point incrementPoint(Point somePoint){
    somePoint.x+=1;
    somePoint.y+=1;
    return somePoint;
};
```

# Passing structures to functions



DISPLAY 10.1 A Structure Definition

```
//Program to demonstrate the CDAccount structure type.
#include <iostream>
using namespace std;
//Structure for a bank certificate of deposit:
struct CDAccount
{
    double balance;
    double interestRate;
    int term; //months until maturity
};


void getData(CDAccount& theAccount);
//Postcondition: theAccount.balance and theAccount.interestRate
//have been given values that the user entered at the keyboard.


int main( )
{
    CDAccount account;
    getData(account);

    double rateFraction, interest;
    rateFraction = account.interestRate / 100.0;
    interest = account.balance * rateFraction * (account.term / 12.0);
    account.balance = account.balance + interest;

    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(2);
    cout << "When your CD matures in "
         << account.term << " months,\n"
         << "it will have a balance of $"
         << account.balance << endl;
    return 0;
}
```

```
//Uses iostream:
void getData(CDAccount& theAccount)
{
    cout << "Enter account balance: $";
    cin >> theAccount.balance;
    cout << "Enter account interest rate: ";
    cin >> theAccount.interestRate;
    cout << "Enter the number of months until maturity\n"
         << "(must be 12 or fewer months): ";
    cin >> theAccount.term;
}
```

from: Problem Solving with C++, 10th Edition, Walter Savitch

# Be careful of same member names

```
// defining the struct
struct Point {
     int x;
     int y;
};

struct Character {
    int x;
    int y;
    std::string name;
};
```

Compiler can keep track but it's harder for humans

# Structs and Pointers

```
struct Books {
   std::string title;
   std::string author;
   std::string subject;
   int book_id;
};
```

# Pointers and Structs

- You can define pointers to structures in very similar way as you define pointer to any other variable

```
struct Books *struct_pointer;
struct Books Book1;
```

- Now, you can store the address of a structure variable in the above defined pointer variable.

```
struct_pointer = &Book1;
```

# Structs and Pointers

```
void printBook( struct Books *book ) {
   std::cout << "Book title : " << (*book).title;
   std::cout << "Book author : " << (*book).author;
   std::cout << "Book subject : " << (*book).subject;
   std::cout << "Book id : " << (*book).book_id;
}
```

```
struct Books {
   std::string title;
   std::string author;
   std::string subject;
   int book_id;
};
```

```
int main(){
    struct Books Book1;

    Book1.title = "Learn C++ Programming"
    Book1.author = "Chand Miyan"
    Book1.subject = "Computer Science"

     printBook( &Book1 );
}
```

# Example

- Write a Student struct that contains
  - Name
  - StudentID
  - Major
- Implement functions:
  - void buildStudent(Student &someStudent)
    - Initialize member variables of student Struct
  - void changeMajor(Student &someStudent);
    - Change the major of a student structure
  - void printStudent(Student &someStudent);
    - Prints out all member variables of student structure