CSE 220 – C Programming
Fall 2015

Structures, Enumerations, Unions
Storage

• Members are stored in memory in the ORDER in which they are declared.

```c
struct Car {
    int x;
    float y;
    double r;
} car1;
```

`car1`
Accessing Struct Members (I)

• To access the value of a struct member use:

```c
struct
{
    type_1 member_1;
    type_2 member_2;
    type_3 member_3;
} structure_name;
```

```c
structure_name.member_1 = ?;
```
Accessing Struct Members (II)

• To access the value of a struct member use:

1. ```
struct structure_tag_name
{
    type_1 member_1;
    type_2 member_2;
    type_3 member_3;
}
;```

2. ```
struct structure_tag_name instance_name;
```

3. ```
instance_name.member_1 = ?;
```
Accessing Struct Members (II)

```c
#include <stdio.h>

struct Circle{
    float x;
    float y;
    float r;
};

int main()
{
    Circle.x = 1.2;
    return 0;
}
```

```c
#include <stdio.h>

struct{
    float x;
    float y;
    float r;
} Circle;

int main()
{
    Circle.x = 1.2;
    printf("Circle.x = %f \n", Circle.x);
    return 0;
}
```
Accessing Struct Members

This program illustrates creating structs and then declaring and using struct variables.
Accessing Struct Members

```
#include <stdio.h>

struct Circle{
    float x;
    float y;
    float r;
};

int main()
{
    struct Circle circle1;
    return 0;
}
```
Accessing Struct Members

```c
#include <stdio.h>

struct Circle{
    float x;
    float y;
    float r;
};

int main()
{
    struct Circle circle1;

    circle1.x = 1.2;
    circle1.y = 2.5;
    circle1.r = 1;

    return 0;
}
```
Accessing Struct Members

```c
#include <stdio.h>

struct Circle{
    float x;
    float y;
    float r;
};

int main()
{
    struct Circle circle1;

    circle1.x = 1.2;
    circle1.y = 2.5;
    circle1.r = 1;

    printf("The circle centered at (%f,%f) with radius %f. \n", circle1.x, circle1.y, circle1.r);

    return 0;
}
```
# Accessing Struct Members

```c
#include <stdio.h>

struct Circle{
    float x;
    float y;
    float r;
};

int main()
{
    struct Circle circle1;

    circle1.x = 1.2;
    circle1.y = 2.5;
    circle1.r = 1;

    printf("The circle centered at (%f,%f) with radius %f. \n", circle1.x, circle1.y, circle1.r);
    return 0;
}
```

```
 gcc struct.cc -o struct.exe
./struct.exe
The circle centered at (1.200000,2.500000) with radius 1.000000.
```
Accessing Struct Members

You can create instances here!
Initialization (I)

- Structures can be initialized at the time of **instantiation**:

```c
struct structure_name instance_name = {value_1, value_2, value_2, ...};
```

- Do not forget the order of members!

- Similar to initialization of arrays!
Initialization (I)

- Structures can be initialized at the time of instantiation:

```c
#include <stdio.h>

struct Circle{
    float x;
    float y;
    float r;
} circle1;

int main()
{
    struct Circle circle1 = {1.2, 2.5, 1};
}
```

Values in initializer **MUST** appear in the same order as in the structure
Initialization (II)

• Structures can also be initialized at the same time of *declaration*:

```c
struct structure_name
{
    type_1 member_1;
    type_2 member_2;
    type_3 member_3;
    ...
} instance_name{value_1, value_2, value_2, ...};
```
Initialization (II)

Values in initializer **MUST** appear in the same order as in the structure
Values in initializer **MUST** appear in the same order as in the structure
Copying Structures

- Only if you defined structure tag in declaration

- Copy a struct into another:
  \[
  \text{circle1} = \text{circle2};
  \]

- copies individual members one by one
  \[ \text{circle1} \text{ and } \text{circle2} \text{ must have compatible types} \]
Copying Structures with Tag

```c
#include <stdio.h>

struct Circle{
    float x;
    float y;
    float r;
} circle1;

int main()
{
    struct Circle circle1 = {1.2, 2.5, 1};
    struct Circle copiedCircle;
    copiedCircle = circle1;

    printf("The copied circle centered at (%f,%f) with radius %f. \n",
           copiedCircle.x, copiedCircle.y, copiedCircle.r);

    return 0;
}
```

```
gcc struct.cc -o struct.exe
./struct.exe
The copied circle centered at (1.200000,2.500000) with radius 1.000000.
cse220:~>
```
Copying Structures without Tag

• Do you think this code works?
Copying Structures without Tag

• Only if you defined structure tag in declaration

We can NOT copy a struct to another struct unless they are instances of a tag!
Copying Arrays

Normally, arrays **cannot** be copied using the = operator, **but when inside a struct, they can!**
Why Structure Tags?

Repeated structure information: program hard to maintain [adding or removing a member to struct]

```c
struct{
  float x;
  float y;
  float r;
} circle1;

struct{
  float x;
  float y;
  float r;
} circle2;

struct Circle { 
  float x;
  float y;
  float r;
};

struct Circle circle1, circle2, circle3;
```
Why Structure Tags?

• The 2 structs are **not** compatible with each others
  – treated as two different types of struct

```c
struct{
    float x;
    float y;
    float r;
} circle1;

struct{
    float x;
    float y;
    float r;
} circle2;
```

```c
struct Circle
{
    float x;
    float y;
    float r;
};
```

```c
struct Circle circle1, circle2;
```

circle 1 = circle2;  // X

circle 1 = circle2;  // ✓
Comparing Structures

- Can **NOT** use `==` or `!=` to check if 2 structs are equal or not
Comparing Structures

We need to compare the members one by one!

Compiler Error!
Structs in Functions

- Structs can be passed as parameters to a function and as return values.

- Structs are passed by value (a copy is made and sent to the function).

```c
void printCarInfo(struct Car car)
{
    printf("%d: %s\n", car.year, car.make);
}

struct Car newerCar(struct Car car1, struct Car car2)
{
    return car1.year > car2.year ? car1 : car2;
}
```
typedef

• Define (re-name) a type

    typedef oldType newTypeName

• E.g.,

    typedef long int largeNumber;

    largeNumber n = 999999999;
typedef

Do not want to put “struct” when you create instances?

There is an easier way to define structs or you could "alias" types you create. For example:

```
typedef struct{
    float x;
    float y;
    float r;
} Circle;
```

Now you can use `Circle` directly to define variables of circle type **without using struct keyword**. Following is the example:

```
Circle circle;
```

use this new data type the same as C's basic data types
typedef

You do not need to use "struct" here
Nested Structures

- A struct can be a member of another struct
- May declare nested structures:

```c
struct Point {
    float x;
    float y;
} ;

struct Circle {
    struct Point center;
    float radius;
} ;
```

```c
struct Circle circle1;
circle1.center.x = 1.2f;
circle1.center.y = 2.5f;
```
Example: Array of Structs

```c
#include <stdio.h>

typedef struct {
    char name[99];
    float GPA;
} Student;

void printInfo(Student[], int);

int main()
{

}

void printInfo(Student students[], int num)
{

}
```
Example: Array of Structs

```c
#include <stdio.h>

typedef struct {
    char name[99];
    float GPA;
} Student;

void printInfo(Student[], int);

int main()
{
    Student cse220_students[100];
    Student honors_students[] = {{"Jim", 4.0}, {"Dalia", 4.0}, {"Katie", 3.9}};
    printInfo(honors_students, 3);
    return 0;
}

void printInfo(Student students[], int num)
{

}
```
Example: Array of Structs

```c
#include <stdio.h>

typedef struct {
    char name[99];
    float GPA;
} Student;

void printInfo(Student[], int);

int main()
{
    Student cse220_students[100];
    Student honors_students[] = { {"Jim", 4.0}, {"Dalia", 4.0}, {"Katie", 3.9} };
    printInfo(honors_students, 3);
    return 0;
}

void printInfo(Student students[], int num)
{
    printf("The list of students:\n");
    for(int i = 0; i < num; i++)
    {
        printf("%d: %s with GPA:%f\n", i+1, students[i].name, students[i].GPA);
    }
    return;
}
```
Example: Array of Structs

```c
#include <stdio.h>

typedef struct {
  char name[50];
  float GPA;
} Student;

void printInfo(Student students[], int num)
{
  printf("The list of students:\n");
  for(int i = 0; i < num; i++)
  {
    printf("%d: %s with GPA:%f\n", i+1, students[i].name, students[i].GPA);
  }
}

void print()
{
  Student students[] = {
    {"Jim", 4.000000},
    {"Dalia", 4.000000},
    {"Katie", 3.900000}
  }
  printInfo(students, 3);
}

int main()
{
  print();
  return 0;
}
```

[cse220:~] gcc -w students.cc -o students.exe
[cse220:~] ./students.exe
The list of students:
1: Jim with GPA:4.000000
2: Dalia with GPA:4.000000
3: Katie with GPA:3.900000
Exercise: Array of Structs

```c
struct Circle {
    float x;
    float y;
    float r;
};
```

Create an array(size=1000) of struct circle and get the members’ values from user.