



Structures II

CS2263 – Systems Software Development

1

Learning Outcomes

At the conclusion of this lecture students should be able to:

- Allocate structures on the heap
- Program with structures containing pointers/other structures
- Construct thematic modules

2

References

- Lu, Yung-Hsiang. 2015. Intermediate C Programming. CRC Press. New York. (Chapter 16)
- Kernighan, Brian, and Dennis Ritchie. 1988. The C Programming Language, Second Edition. Prentice Hall. Upper Saddle River, NJ. (Chapter 6)

3

Same old, Same old

```
typedef struct point2d {  
    double x;  
    double y;  
} Point2D;
```

Memory for pointers to structures must be initialized in same way as for other pointers

- using address of another structure
- using dynamic memory allocation

4

Address of Another Structure

```
typedef struct point {  
    double x;  
    double y;  
} Point2D;  
  
Point2D pt; // stack allocation  
Point2D* pPt = &pt;
```

5

Dynamic Memory Allocation

```
typedef struct point {  
    double x;  
    double y;  
} Point2D;  
  
Point2D* pPt;  
pPt = (Point2D*)malloc(sizeof(Point2D));  
if( (Point2D*)NULL == pPt) {...}
```

6

Accessing Structure Elements

- Access of elements through the structure itself
`Point2D pt; // stack allocation`
`pt.x = 2.5;`

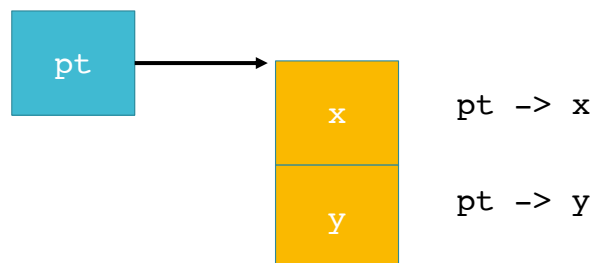
7

Accessing Structure Elements

- Access of elements through a pointer to the structure

```
Point2D* pPt;  
pPt = (Point2D*) malloc(sizeof(Point2D)); //Eeek!
```

```
pPt->x = 2.5;
```



8

Pointer Data Types

- Recall

```
typedef char* String;
```
- It is possible to declare a type as a pointer to another type

```
typedef struct point2D {
    double x;
    double y;
} Point2D, *pPoint2D;
pPoint2D q; // same as Point2D* q
```

9

Structures and Functions

```
Point2D createPoint2D(double x, double y) {
    Point2D ptThis;
    ptThis.x = x;
    ptThis.y = y;
    return ptThis;
}

//Print the contents of the struct - see last day
void printfPoint2D(Point2D ptThis);

// Returns the values that make up the struct
Point2D w = createPoint2D(1, 2.2);
```

10

Structures and Functions

- Passing stuff around: what's the overhead?
 - Structures as parameters
 - Structures as return values
- Stack implications
 - Stack frame size
 - Time to construct new stack frame
- How to get around this?
 - Pointers to structures (pass by reference)

11

Populating a Structure

```
void setPoint2D(pPoint2D pPtThis, double x, double y){
    pPtThis->x = x;
    pPtThis->y = y;

    return;
}
```

Recall } Point2D, *pPoint2D;
 pPoint2D q; // same as Point2D* q

12

Initiating a Structure

```
Point2D* mallocPoint2D() {  
    Point2D* pPtThis;  
    pPtThis = (Point2D*) malloc(sizeof(Point2D));  
  
    return pPtThis;  
}
```

13

Creating a Structure

```
pPoint2D createPoint2D(double x, double y) {  
    pPoint2D pPtThis;  
    pPtThis = mallocPoint2D();  
    if(pPtThis == NULL) return pPtThis;  
  
    fillPoint2D(pPtThis, x, y);  
  
    return pPtThis;  
}
```

14

Comparing Structures

```
int comparePoint2D(const pPoint2D pPtThis, const pPoint2D
pPtThat) {
    return pPtThis->x == pPtThat->x && pPtThis->y == pPtThat->y;
}
```

- Why is this a bad idea?

```
int same = comparePoint2D(pPtA, createPoint2D(3, 7) );
```

Recall

```
Point2D createPoint2D(double x, double y) {
```

15

Arrays of Structures (I)

```
Point2D rect [4];

pPoint2D rectangle;
pPoint2D corner;
double x, y;
int iNRead;

rectangle = malloc(4*sizeof(Point2D) );
if(rectangle == (Point2D*)NULL){...}

for(int i=0; i<4; i++){
    printf("Enter two values:");
    iNRead = scanf("%lf %lf", &x, &y);
    if(iNRead != 2){...} /* error */

    fillPoint2D(&rectangle[i], x, y);
}
```

16

Arrays of Structures (II)

```
int i;
for(i=0; i < 4; i++){
    printf("vertex %d = (%f %f)\n", i,
        rectangle[i].x, rectangle[i].y);
}
```

17

Structures of Arrays

```
#define MAX 20
typedef struct {
    char name[MAX+1];
    Point2D vertices[4];
} Rectangle, *pRectangle;
```

```
RectangleT r;
pRectangle pr
```

- Assume the **r** is populated. How do we reference the component parts?
- Assume the **pr** is populated. How do we reference the component parts?

18

With Great
Power Comes
Great
Responsibility

```
//Person.h
#ifndef PERSON_H
#define PERSON_H
typedef struct{
    int year;
    int month;
    int date;
    char* name;
} Person;
Person* constructPerson(char* n, int y, int m, int d);
void freePerson(Person* p);
void printfPerson(Person* p);
#endif
```

Considerations/implications for

- Constructor function
- Copy constructor function
- Destructor function