

Computer Programming 143 – Lecture 23

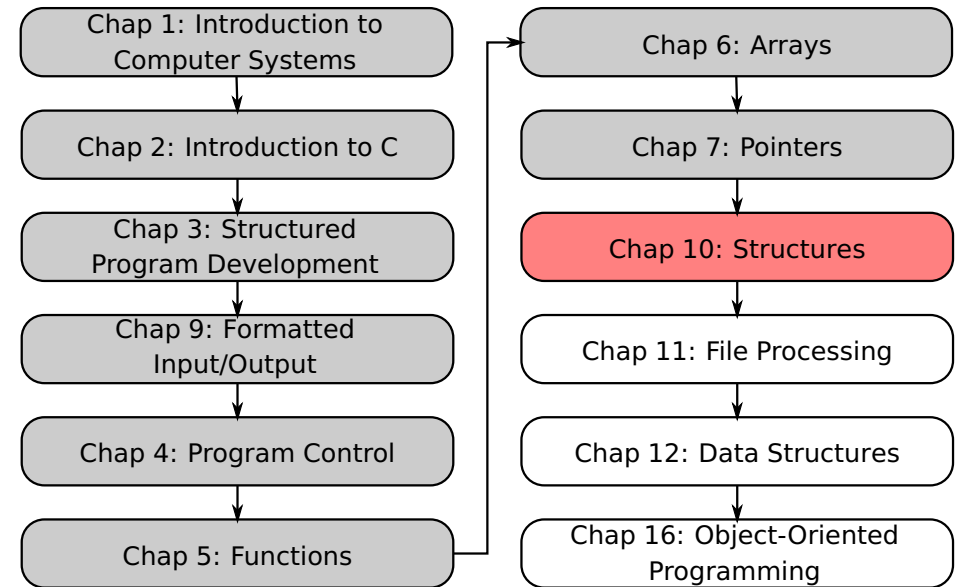
C Structures I

Electrical and Electronic Engineering Department
University of Stellenbosch

Corné van Daalen
Thinus Booysen
Willie Smit
Willem Jordaan



Module Overview



Lecture Overview

- 1 10.1 Structures
- 2 10.2 Structure Definitions
- 3 10.3 Initialising Structures
- 4 10.4 Accessing Members of Structures
- 5 10.5 Using Structures with Functions
- 6 10.6 typedef

10.1 Structures

Structures

- Definition: collections of related variables (aggregates) under one name

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

- Variables in a structure are called *members* of the structure
 - Can contain members of different data types
- Commonly used to define records to be stored in files (Chap. 11)
- Combined with pointers, can create linked lists, stacks, queues, and trees (Chap. 12)

10.2 Structure Definitions I

Example I

```
struct coord {  
    double x;  
    double y;  
}; /*<--- NB ; !! */
```

- **struct** introduces the definition for structure **coord**
- coord is the structure name and is used to declare variables of the structure type
- coord contains two members of type **double**
 - These members are x and y

10.2 Structure Definitions II

Example II

```
struct student {  
    char Name[ 30 ];  
    char gender;  
    long number;  
};
```

Example III

```
struct node {  
    struct node *leftPtr;  
    struct node *rightPtr;  
};
```

struct information

- A struct cannot contain a member that is an instance of itself
- Can contain a member that is a pointer to the same structure type (link lists, etc. – Chap. 12)
- A structure definition does not reserve space in memory
 - Instead creates a new data type used to define structure variables

10.2 Structure Definitions III

Valid Operations

```
struct coord {  
    double x;  
    double y;  
};  
struct coord lCorner, origin = {0.0, 0.0};  
struct coord *cPtr;
```

- 1 Assigning a structure variable to a structure variable of the same type
lCorner = origin;
- 2 Taking the address (&) of a structure variable
cPtr = &lCorner;
- 3 Accessing the members of a structure variable (next sections)
- 4 Using the sizeof operator to determine the size of a structure variable

10.3 Initialising Structures

Example

```
struct coord origin = {0.0, 0.0};
```

Example

```
struct person {  
    char firstName[ 20 ];  
    char lastName[ 20 ];  
    int age;  
    char gender;  
};  
  
struct person DEls = {"Danie", "Els", 22, 'M'};
```

10.4 Accessing Members of Structures

- Dot operator (.) used to access members by means of the *structure name*
- Arrow operator (->) used to access members by means of a *pointer to the structure* (no spaces - >)

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

struct coord c, *cPtr;

cPtr = &c;

/* The following assignments are all equivalent */
c.x = 10.5;           c.y = 12.3;
(*cPtr).x = 10.5;     (*cPtr).y = 12.3;
cPtr->x = 10.5;        cPtr->y = 12.3;
```

Operator precedence

Operator:	Associativity	Type
[] () . ->	left to right	highest
- + ++ -- ! * & (type)	right to left	unary
* / %	left to right	multiplicative
+ -	left to right	additive
< <= > >=	left to right	relational
== !=	left to right	equality
&&	left to right	logical and
	left to right	logical or
?:	right to left	conditional
= += -= *= /= %=	right to left	assignment
,	left to right	comma

Note: `++C.x` is `C.x += 1` is `C.x = C.x + 1`

10.5 Using Structures with Functions

```
#include <stdio.h>

struct coord {
    double x;
    double y;
};

struct coord addCoordVal( struct coord a, struct coord b );
void addCoordRef( const struct coord *aPtr,
                  const struct coord *bPtr, struct coord *rPtr );

int main ()
{
    struct coord pr;
    struct coord pa = { 10.0, 11.0 };
    struct coord pb = { 1.0, 2.0 };

    printf ( "pa:   x = %4.1f, y = %4.1f\n", pa.x, pa.y );
    printf ( "pb:   x = %4.1f, y = %4.1f\n", pb.x, pb.y );
}
```

```
pr = addCoordVal( pa, pb );
printf( "pa+pb: x = %4.1f, y = %4.1f\n", pr.x, pr.y );

addCoordRef( &pa, &pb, &pr );
printf( "pa+pb: x = %4.1f, y = %4.1f\n", pr.x, pr.y );

return 0;
}

/* Call by value and return via return value */
struct coord addCoordVal( struct coord a, struct coord b )
{
    struct coord res;

    res.x = a.x + b.x;
    res.y = a.y + b.y;

    return res;
}
```

```

/* Call by reference and return via argument list */
void addCoordRef( const struct coord *aPtr,
                  const struct coord *bPtr, struct coord *rPtr )
{
    rPtr->x = aPtr->x + bPtr->x;
    rPtr->y = aPtr->y + bPtr->y;
}

```

Output:

```

pa:    x = 10.0, y = 11.0
pb:    x =  1.0, y =  2.0
pa+pb: x = 11.0, y = 13.0
pa+pb: x = 11.0, y = 13.0

```

```

/* typedefExample.c
 * Example using typedef */
#include <stdio.h>

typedef int *IntPtrType;

void printInteger( IntPtrType myPtr );

int main( void )
{
    int a = 5;
    IntPtrType IntPtr = &a; // declares IntPtr as type IntPtr
    printInteger( IntPtr );

    return 0; // indicates successful termination
} // end main

void printInteger( IntPtrType myPtr )
{
    printf( "The value is: %d\n", *myPtr );
} // end function printInteger

```

10.6 typedef

typedef

- Creates synonyms (aliases) for previously defined data types
- Use typedef to create shorter type names
- Example:

*/*Option 1*/*

```

struct coord {
    double x;
    double y;
};

```

```
typedef struct coord Point;
```

```
Point pa, pb;
```

*/*Option 2*/*

```

typedef struct {
    double x;
    double y;
} Point;

```

```
Point pa, pb;
```

- Defines a new type name Point as a synonym for type struct coord
- typedef does not create a new data type, only an alias

Perspective

Today

C Structures

- Introduction to structures
- Structure definitions
- Initialising structures
- Accessing structure members
- Using structures with functions
- typedef

Next lecture

- Arrays of Structures
- malloc() and Structures
- Example: shuffling and dealing of cards
- Enumeration constants

Homework

- 1 Study Sections 10.1-10.6 in Deitel & Deitel
- 2 Do Self Review Exercises 10.2(a),(c)-(g), 10.3, 10.4(a),(b),(d)-(f) in Deitel & Deitel
- 3 Do Exercises 10.6 in Deitel & Deitel