Program Patterns: struct

J.W. Choi

2024

Structure

Basic Features of C

- variable and type declarations, expressions
- assignment
- if-else, if-else chain, switch
- for and while loop
- functions (call by value)
- pointers
- functions (call by reference)
- data structures (array, struct, linked lists, stack, queue, tree, graph)
- input & output (monitor, hard disk drive)

Structure

- Array: collection of same types of data
- Structure: collection of different types of data

How to define and store related information?

```
char   name[20];
int    age;
float  salary;
char   hobby[3][20];
```

for the same person

C struct

```
struct {
     char   name[20];
     int    age;
     float  salary;
     char   hobby[3][20];
}  employee;

/*  name, age, salary, hobby:  members      */
/*  each member is a variable               */
/*  employee:   variable of type  struct { } */
```

C struct

```
struct {
    char   name[20];
    int    age;
    float  salary;
    char   hobby[3][20];
}  employee;
```

same as

```
struct {char name[20]; int age; float  salary;
    char hobby[3][20];}  employee;
```

Structure tag name

```
struct  EMPRECORD {
     char   name[20];
     int      age;
     float   salary;
     char   hobby[3][20];
};

/*  EMPRECORD:  tag name for { }          */

/*  struct  EMPRECORD
               employee, former_employee;   */
```

Structure

- “data_type” “variable”
- struct {member declarations} variable
  - Each member is a variable.
- struct tag_name variable
- typedef struct tag_name struct_name
- typedef struct {member declarations} struct_name

Member Name Scope

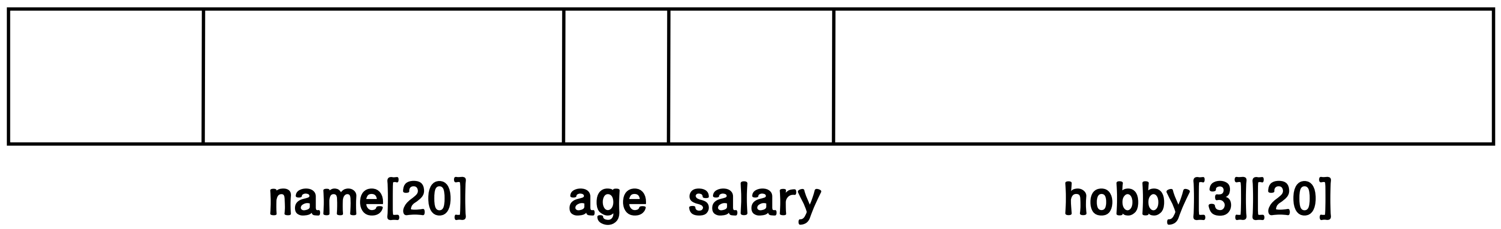
```
struct  {
    char   name[20];
    int      age;
    float   salary;
    char   hobby[3][20];
}    employee;

struct  {
    char   name[20];
    int      age;
    char   address[30]'
}    person;
```

/* unique only within a single structure   */

Memory Allocation (Contiguous Space)

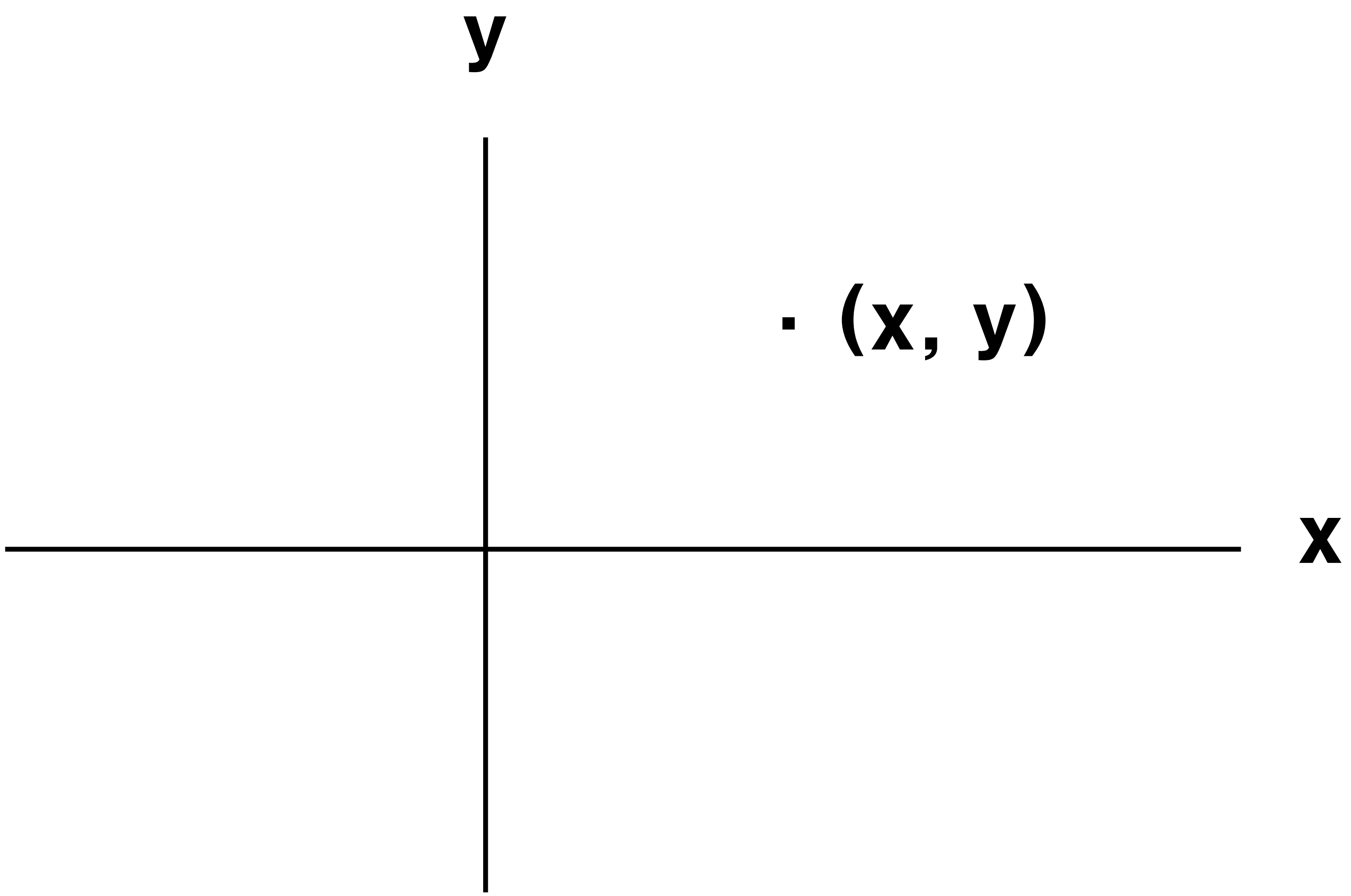
```
struct {
    char   name[20];
    int    age;
    float  salary;
    char   hobby[3][20];
}  employee;
```



Member Data Types

- Primitive types
  - int, float, double, char
  - pointer
- Array
- Structure
  - other struct
  - defining struct

Exercise: Define a struct for a Point



Solution

```
 struct
{
    int     x, y;
 } point;
```

Structure Member Access

- Variable Name . Member Name
  - struct_variable.member_name

Example

```
struct {
    char   name[20];
    int    age;
    float  salary;
    char   hobby[3][20];
} employee;

/*  employee.name           */
/*  employee.hobby[2]   */
```

Structure Initialization

- Initialize each structure member
  - struct_variable.member_name = expression;
- Initialize the entire structure
  - struct_variable = expression;
  - Each element of the expression is assigned to each corresponding member of the structure.

Example: Member of a struct

```
struct EMPRECORD {
    char   name[20];
    int    age;
    float  salary;
    char   hobby[3][20];
} employee;

strcpy( employee.name, “Neil Diamond” );
strcpy( employee.hobby[2], “tennis and walking” );
```

Example: Entire struct

```
struct EMPRECORD {
     char   name[20];
     int      age;
     float   salary;
     char   hobby[3][20];
} employee = {“hong gildong”, 25, 35000.0, “jump”};
```

or

```
={.age=25, .name=“hong gildong”, };
```

Structure Within a Structure

- Struct containing other struct
  - "other struct" must be defined first.
- Struct containing the same struct
  - "self referential" struct

Member of Struct {} Type

```
struct CAR {
     char make[20];
     char model[20];
     int  year;
 }  car;

struct {
     char name[20];
     int  age;
     struct CAR car_owned;
 }  employee;
```

Member Access

```
struct CAR {
      char make[20];
      char model[20];
      int  year;
  }  car;

  struct  {
      char name[20];
      int  age;
      struct CAR car_owned;
  }  employee;

 /* employee.car_owned.model  */
```

Exercises (Textbook Chapter 12)

- Programming Exercise 12.1 1a
  - Read the month, day, year.
  - Store them in a struct.
  - Print the month, day, year.
- Programming Exercise 12.1 2
  - Read the company name (char[20]), stock earnings per share (float), price to earnings ratio (float).
  - Store them in a struct.
  - Calculate the stock price (earnings per share * price to earnings ratio).
  - Print the company name and stock price
  - Repeat 5 times with different data.

Programming Exercises

- 12.1 1a. Write a C program that prompts a user to input the current month, day, and year. Store the data entered in a suitably defined structure and display the date in an appropriate manner.
- 12.1 2. Write a program that uses a structure for storing the name of a stock, its estimated earnings per share, and its estimated price-to-earnings ratio.
- Have the program prompt the user to enter these items for five different stocks, each time using the same structure to store the entered data.
- When the data have been entered for a particular stock, have the program compute and display the anticipated stock price based on the entered earnings and price-per-earnings values.
- For example, if a user entered the data XYZ 1.56 12, the anticipated price for a share of XYZ stock is (1.56)*(12) = $18.72.

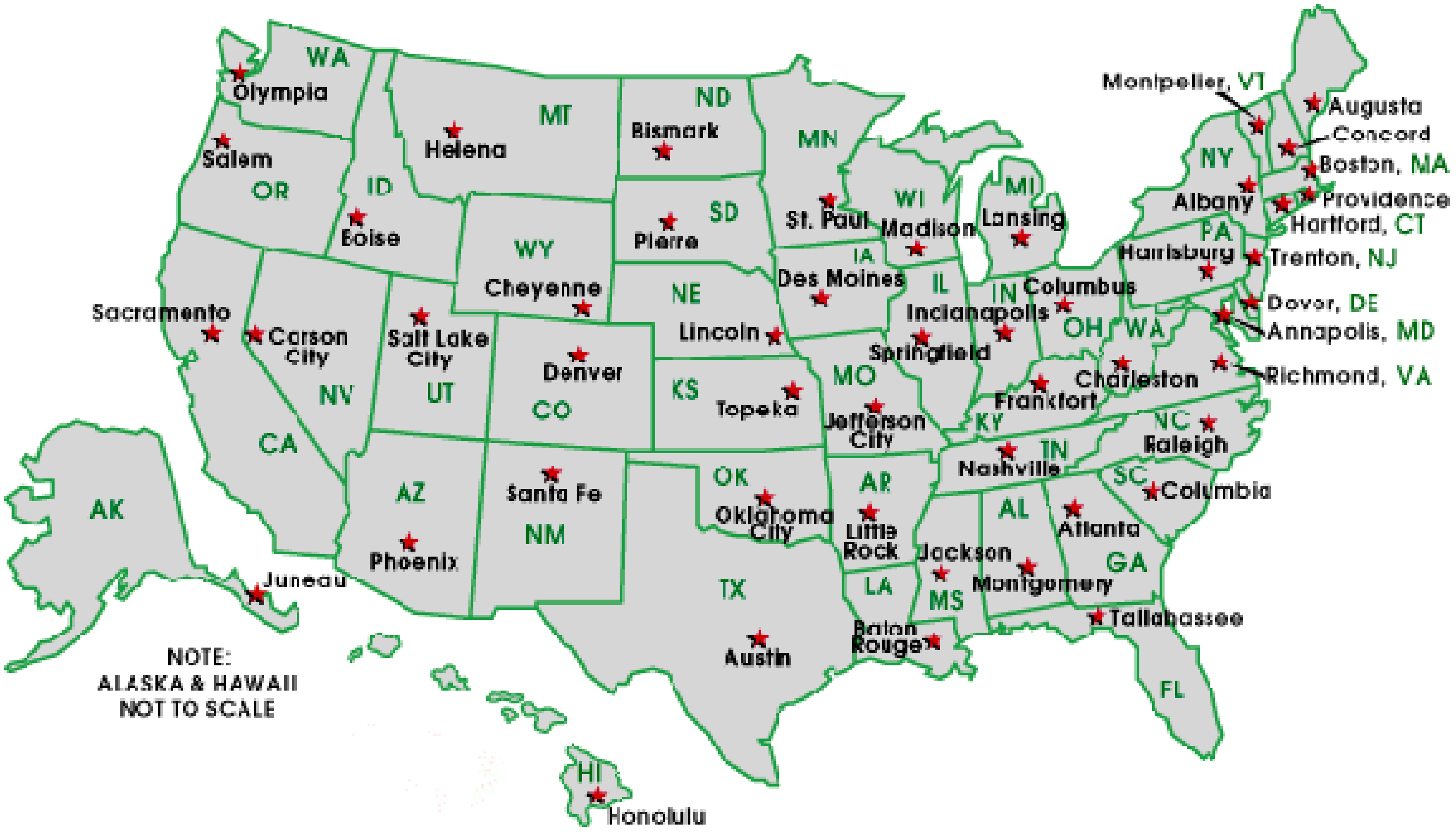
Array of Structures

Array of Structures

- "data type" "array_name" "["max_size"]"
- struct EMPRECORD employee[250];

- Real power of structures
  - combined with looping, if-else (switch), functions
  - search, sort, compute, display,...

50 States of the United States



Array of Structs

```
struct   {
     char   name[20];
     char   abbr[2];
     int     population;
     float   area;
     double GDP;
     char   governor[20];
} US_States[50];
```

Using the Array of structs

- Find the state with the highest population
- Find 5 states with the highest GDP
- Find states with the same first character
- Sort states in alphabetical order of names
- …

Array of Structs

```
struct {
    char   name[20];
    int    age;
    float  salary;
    char   hobby[3][20];
} employee[300];
```

Table of Employees

name	age	salary	hobby
Kim	25	200	basketball
Lee	30	300	swimming
Park	23	250	music, soccer
Cho	40	500	sleeping
…			
Chung	28	900	game, cooking

Using an Array of structs

- Find the 3 highest-paid employees
- Find employees who like basketball
- Find all employees named Kim
- Sort employees in the age order
- …

Exercises (Textbook Chapter 12)

- Programming Exercise 12.2 1
  - Define an array of 4 structs
    - struct MONTH_DAYS

```
{
   char month_name[10];
   int   days;
 }
```

  - Initialize the array (with your own data).
  - Print the name and days of each month.

Programming Exercise

- 12.2 1. Using the following declaration

```
struct MonthDays
{
    char  name[10];
    int  days;
 }
```

define an array of 12 structures of type MonthDays. Name the array convert[], and initialize the array with the names of the 12 months in a year and the number of days in each month.

- Include the array in a program that displays the name and number of days in each month.

Exercises (Textbook Chapter 12)

- Programming Exercise 12.2   3
  - Define an array of 6 structs for employees.
    - struct has 4 members: last name (char[20]), ID (int), pay_rate (float), hours_worked (float)
  - Initialize each struct with data (given in the book).
  - Calculate the total pay for each employee.
  - Print the name, ID, and total pay for each employee.

Programming Exercise

- 12.2 3a. Declare a single structure type suitable for an employee record consisting of an integer identification number, a last name (consisting of a maximum of 20 characters), a floating-point pay rate, and a floating-point number of hours worked.
- 12.2 3b. Using the structure, write a C program that interactively accepts the following data into an array of six structures:

ID Number	Name	Pay Rate	Hours Worked
3462	Jones	4.62	40.0
6793	Robbins	5.83	38.5
6985	Smith	5.22	45.5
7834	Swain	6.89	40.0
8867	Timmins	6.43	35.5
9002	Williams	4.75	42.0

- Once the data have been entered, the program should create a payroll report listing each employee's name, number, and gross pay. Include the total gross pay of all employees at the end of the report.

struct with a Pointer Member

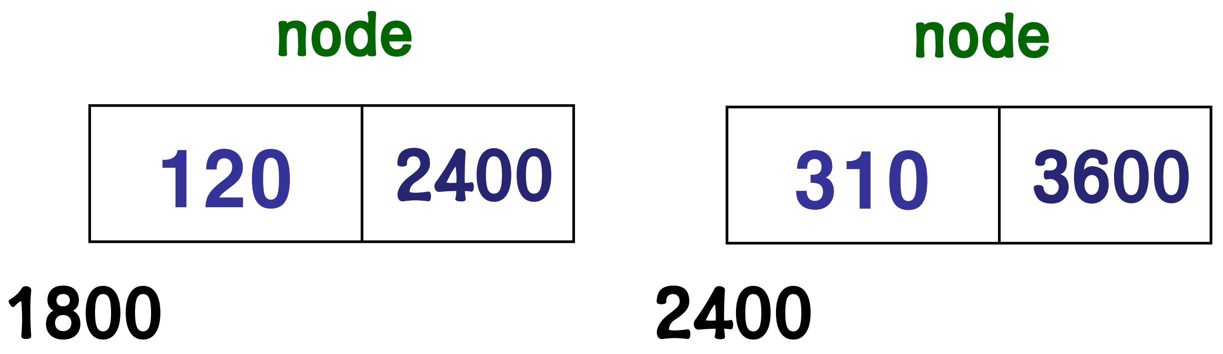
```
struct {
      char    name[20];
      int     age;
      float   GPA;
      int     *grade_ptr;
};
```

Example: Defining a Data Node (1/2)

```
struct  NODE {
        int                         key;
        struct NODE      *next;
};
```

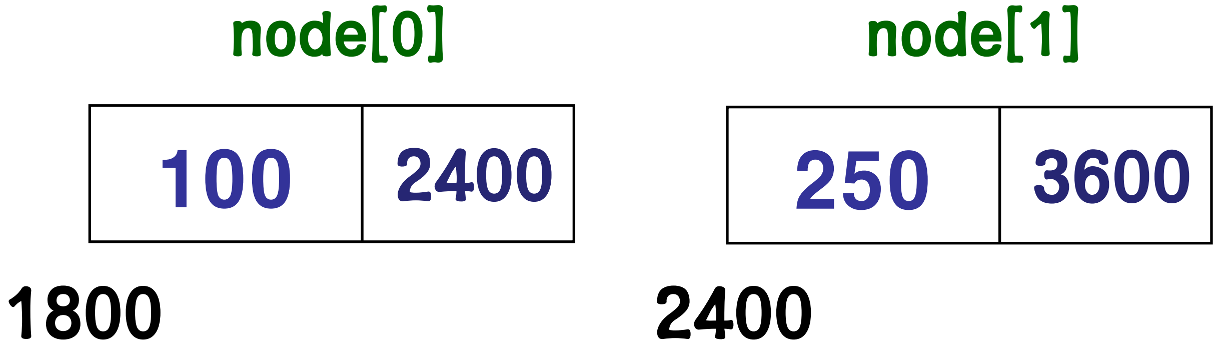
Example: Defining a Data Node (2/2)

```
struct  NODE {
    int                  key;
    struct NODE      *next;
};
```



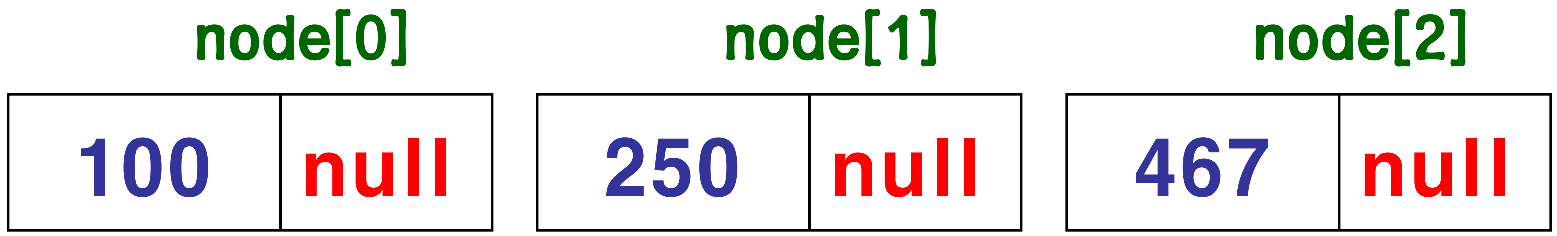
Defining a Data Node Array

```
struct  NODE {
    int                 key;
    struct NODE     *next;
} node[3];
```



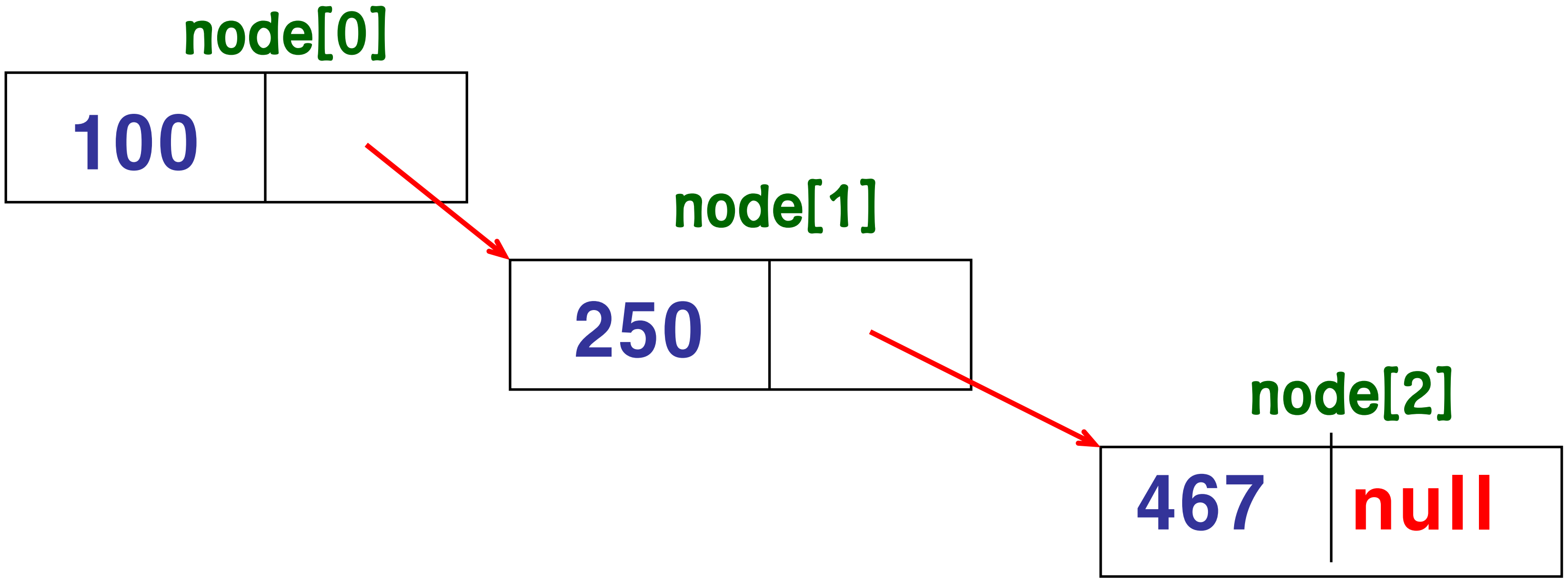
Creating (Filling) a Data Node Array

```
node[0].key = 100;
node[1].key = 250;
node[2].key = 467;
node[0].next = node[1].next = node[2].next = NULL;
```



Linking the Data Nodes (linked list data structure -- later)

```
node[0].next = &node[1];
node[1].next = &node[2];
```



Exercise

- Define an array of 5 structs, and store any string as a key in each node. Then link the five nodes.

Structures and Functions

Function Call with a struct as an Argument

- Call by Value
  - pass a copy of a structure
  - pass a copy of a member of a structure
  - Function may return a structure or a member of a structure.
- Call by Reference
  - Explicitly pass the address of a structure. (unlike passing an array)
  - Function may make changes to the structure.

Function Call with a struct as an Argument

- Define struct data type globally
  - for it to be known to the function being called

Call by Value: (single struct)

```
struct EMPRECORD {
    char  name[20];
    int   age;
    float  salary;
    char  hobby[3][20];
} employee, newemp;


…
new_emp = update_records (employee);  /* function call */

-----------------------

struct EMPRECORD   update_records  /* function definition */
       (struct EMPRECORD  emp) {
            emp.age = 25;
            return emp;
       }
```

Call by Reference: (single struct)

```
struct  EMPRECORD {
    char   name[20];
    int    age;
    float  salary;
    char   hobby[3][20];
} employee;

...
update_records (&employee);     /* function call  */

-----------------------
void   update_records      /* function definition  */
       (struct EMPRECORD  *emp) {
              (*emp).age  =  25;
        }
```

Struct and Pointer

- (*emp).age vs. *emp.age
  - different
- (*emp).age vs. emp -> age
  - same

Call by Reference: (array of structs)

```
struct  EMPRECORD {
    char   name[20];
    int     age;
    float  salary;
    char   hobby[3][20];
} employee[300];


...
update_records (employee);  /* function call */

----------------------
void   update_records   /* function definition */
        (struct EMPRECORD  emp[]) {
                (emp[100]).age = 25;
         }
```

Homework

- Chapter 12 Programming Exercises
  - 12.2 #4 (5 points)
  - 12.3 #1, 3, 4 (total 15 points)

Programming Exercises

- 12.2 4a Declare a single structure type suitable for a car record consisting of an integer car identification number, an integer value for the miles driven by the car, and an integer value for the number of gallons used by each car.
- 12.2 4b Using the structure, write a C program that interactively accepts the following data into an array of five structures:

Car Number	Miles Driven	Gallons Used
25	1,450	62
36	3,240	136
44	1,792	76
52	2,360	105
68	2,114	67

- Once the data have been entered, the program should create a report listing each car number and the miles per gallon achieved by the car. At the end of the report, include the average miles per gallon achieved by the five cars.

Programming Exercises

- 12.3 1. Write a C function named Days() that determines the number of days from the date 1/1/2000 for any date passed as a structure. Use the following Date structure:

```
struct Date
{
    int  month;
    int  day;
    int year;
 }
```

In writing the Days() function, assume that all years have 360 days and each month has 30 days. The function should return the number of days for any date structure passed to it.

Programming Exercises

- 12.3 3. Rewrite the Days() function so that it directly accesses a Date structure, as opposed to receiving a copy of the structure.
- 12.3 4a Write a C function named recent() that returns the later date of any two dates passed to it. For example, if the dates 10/9/2001 and 11/3/2001 are passed to recent(), the second date would be returned.
- 12.3 4b Include the recent() function in a complete program. Store the data structure returned by recent() in a separate date structure and display the member values of the returned date.