Program Patterns:
struct, Search Patterns

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

```
struct {
    char   name[20];
    int    age;
    float  salary;
    char   hobby[3][10];
}  employee;
```

```
int a;
struct employee;
```

Memory Allocation (Contiguous Space)

```
struct {
    char   name[20];
    int    age;
    float  salary;
    char   hobby[3][10];
}  employee;
```

	20	4	4	30
	name[20]	**age**	**salary**	**hobby[3][20]**

22 + 4 + 4 + 60 = 58 -> sizeof(emplyoee) 60

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

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 */

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”};
```

Member Data Types

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

Use of struct employee

```
// type1
struct employee {
        char   name[20];
        int     age;
        float   salary;
        char   hobby[3][20];
  }

 int main(void){
   struct employee per1, per2;
   per1.age = 20;
 …
 …
 }
```

```
// ver2
 struct {
        char   name[20];
        int     age;
        float   salary;
        char   hobby[3][20];
  } employee;

 int main(void){
 employee.age= 20;
 …
 }
```

```
// type3  : use of typedef
struct Employee
        char  name[20];
        int    age; {
        float  salary;
        char  hobby[3][20];
  }

 int main(void){
    typedef struct Employee employee;
    employee em1, em2;
    em1.age = 20;
    em2.age = 30;
 …
 …
 }
```

struct Employee

name	age	salar	Hobby[0]	Hobby[1]	Hobby[3]

struct Employee $\xrightarrow{\text{rename}}$ employee

em1

name	age	salar	Hobby[0]	Hobby[1]	Hobby[3]

name	age	salar	Hobby[0]	Hobby[1]	Hobby[3]

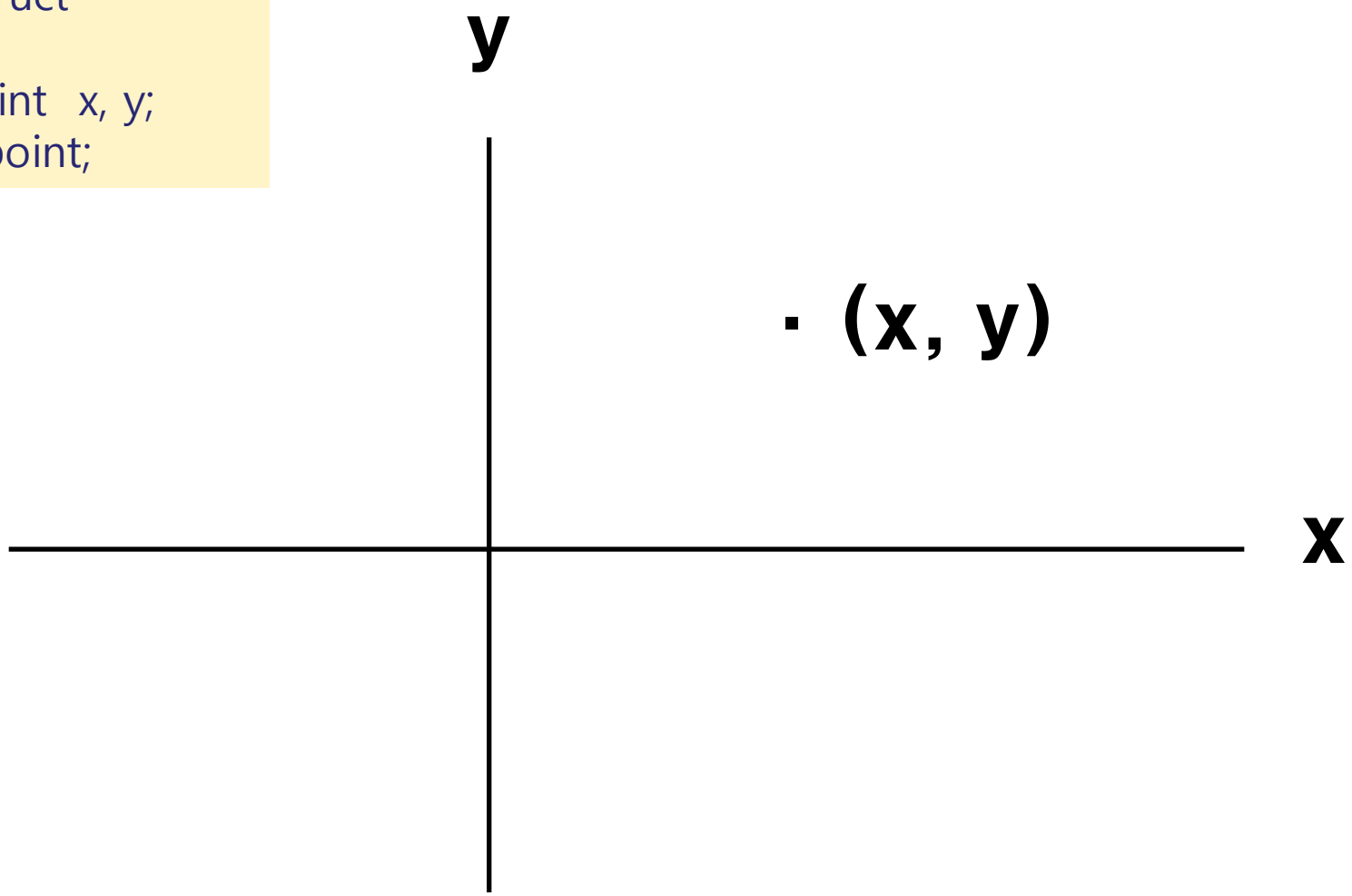
em2

```
//Ver 1
int main() {
    strcpy(em1.name, "Hong Gil Dong");
    em1.age = 10;
    em1.salary = 800000;
    strcpy(em1.hobby[0], "Tennis");
    strcpy(em1.hobby[1], "Sleeping");

    strcpy(em2.name, "Kang Gam Chan");
    em2.age = 40;
    em2.salary = 900000;
    strcpy(em1.hobby[0], "Running");

    printf("em1.name = %s \ n", em1.name);
    printf("em1.age = %d \ n", em1.age);
    printf("em1.salary = %.1f \ n", em1.salary);
    printf("em1.hobby = %s \ n", em1.hobby[0]);
}
```

```
//Ver 1
int main() {
    em1 = { "Hong Gil Dong", 30, 8000000, "Tennis" };
    em2 = { "Kang Gil Dong", 20, 9000000, "Tennis" };

    printf("em1.name = %s \ n", em1.name);
    printf("em1.age = %d \ n", em1.age);
    printf("em1.salary = %.1f \ n", em1.salary);
    printf("em1.hobby = %s \ n", em1.hobby[0]);
}
```

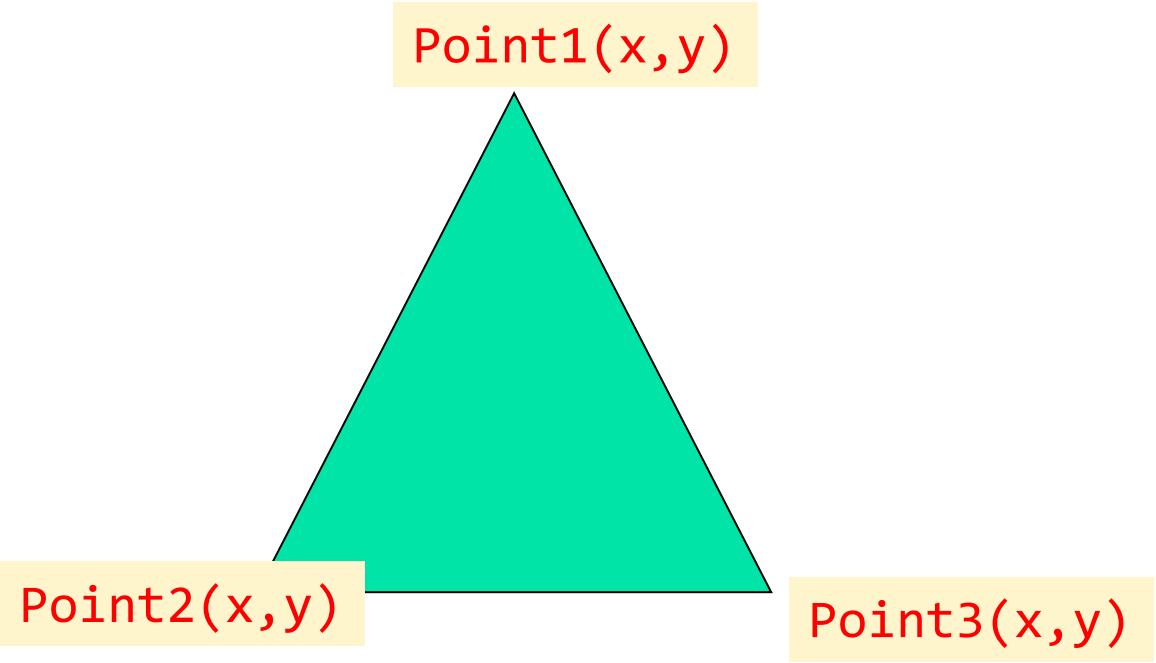
Exercise: Define a struct for a Point

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



Solution

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

struct triangle
{
    struct point point1;
    struct point point2;
    struct point point3;
};

struct triangle triangle1;
triangle1.point1.x = 10;
triangle1.point1.y = 20;
```



Exercise

```c
#include <stdio.h>

struct point
{
   int x;
   int y;
};

struct triangle
{
   struct point point1;
   struct point point2;
   struct point point3;
};
```

```c
int main(void)
{
   struct triangle triangle1 = { { 1, 1 }, { 10, 1 }, { 5, 5 } };

   printf("triangle1.point1.x = %d \n", triangle1.point1.x);
   printf("triangle1.point1.y = %d \n", triangle1.point1.y);
   printf("triangle1.point2.x = %d \n", triangle1.point2.x);
   printf("triangle1.point2.y = %d \n", triangle1.point2.y);
   printf("triangle1.point3.x = %d \n", triangle1.point3.x);
   printf("triangle1.point3.y = %d \n", triangle1.point3.y);

   return 0;
}
```

End of Class

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.