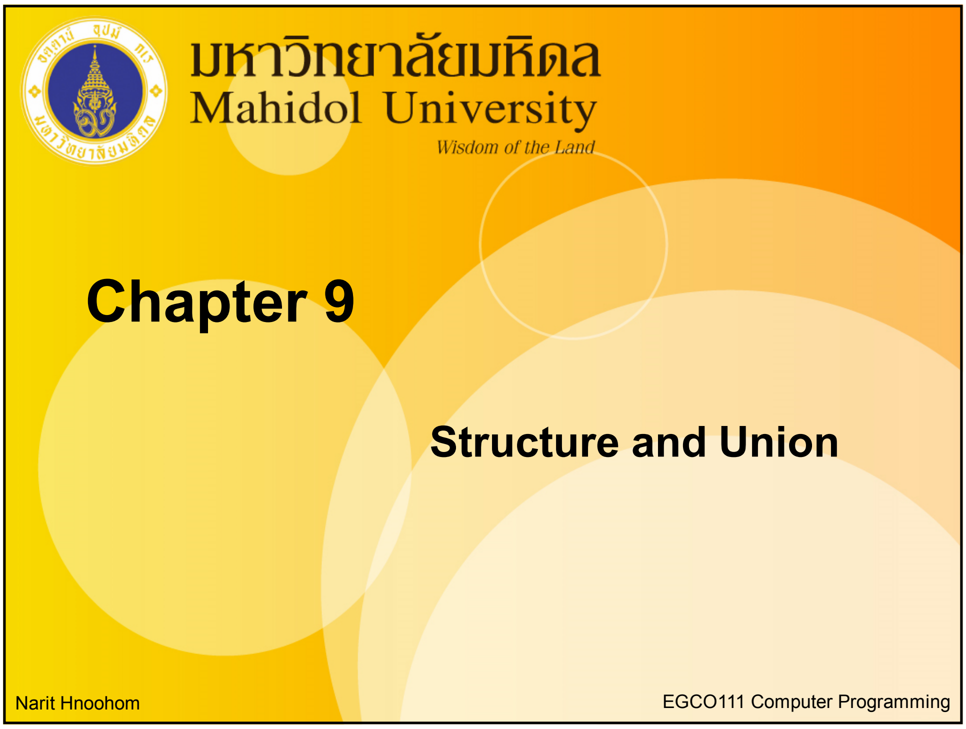


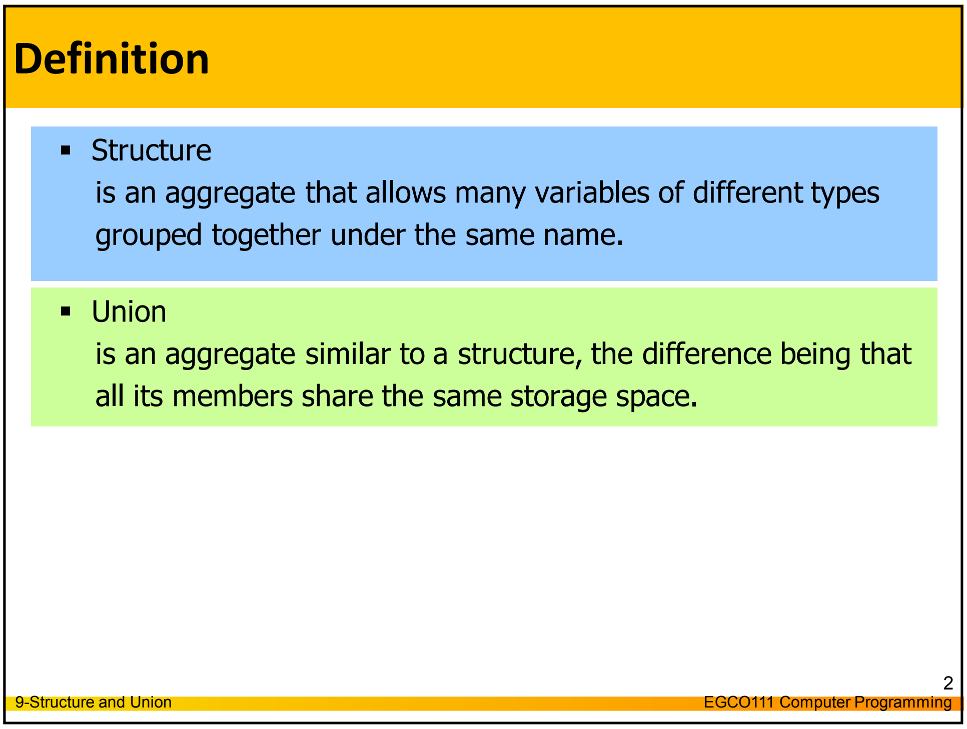
มหาวิทยาลัยมหิดล
Mahidol University
Wisdom of the Land

Chapter 9

Structure and Union

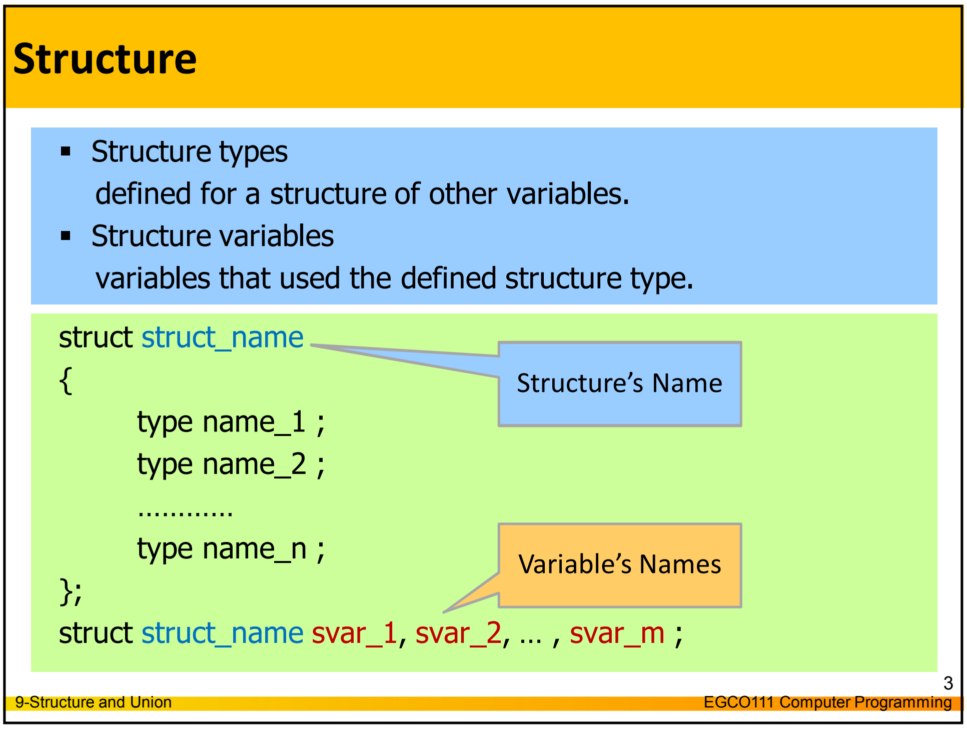
Narit Hnoohom

EGCO111 Computer Programming



Definition

- Structure
  is an aggregate that allows many variables of different types grouped together under the same name.
- Union
  is an aggregate similar to a structure, the difference being that all its members share the same storage space.



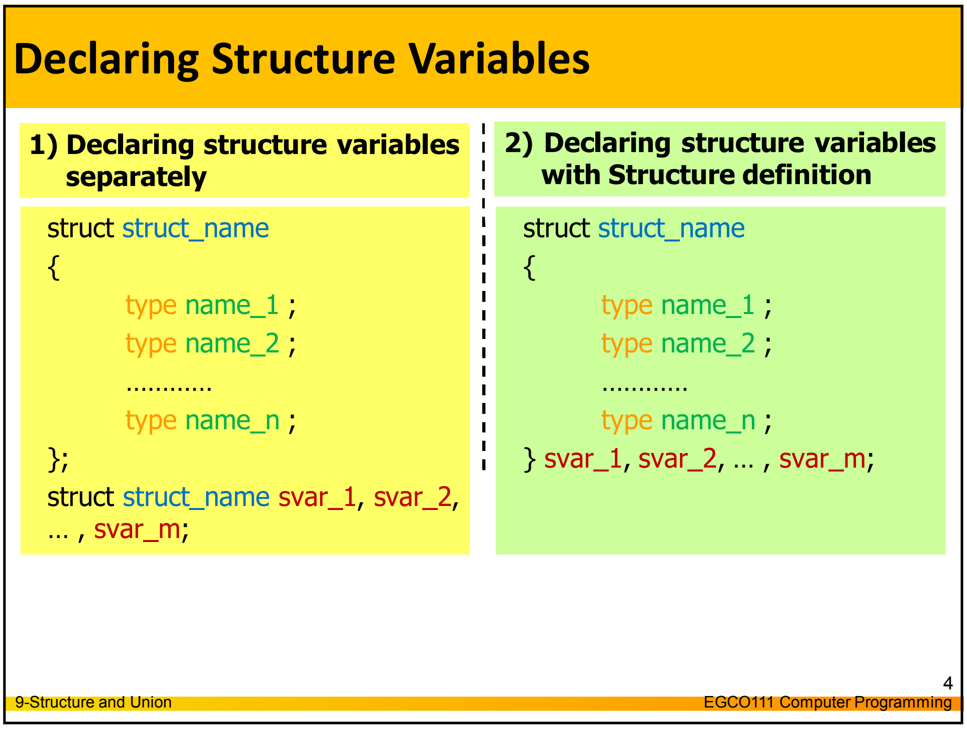
Structure

- Structure types
  defined for a structure of other variables.
- Structure variables
  variables that used the defined structure type.

```
struct struct_name
{
        type name_1 ;
        type name_2 ;
        ...........
        type name_n ;
};
struct struct_name svar_1, svar_2, … , svar_m ;
```

Structure’s Name

Variable’s Names



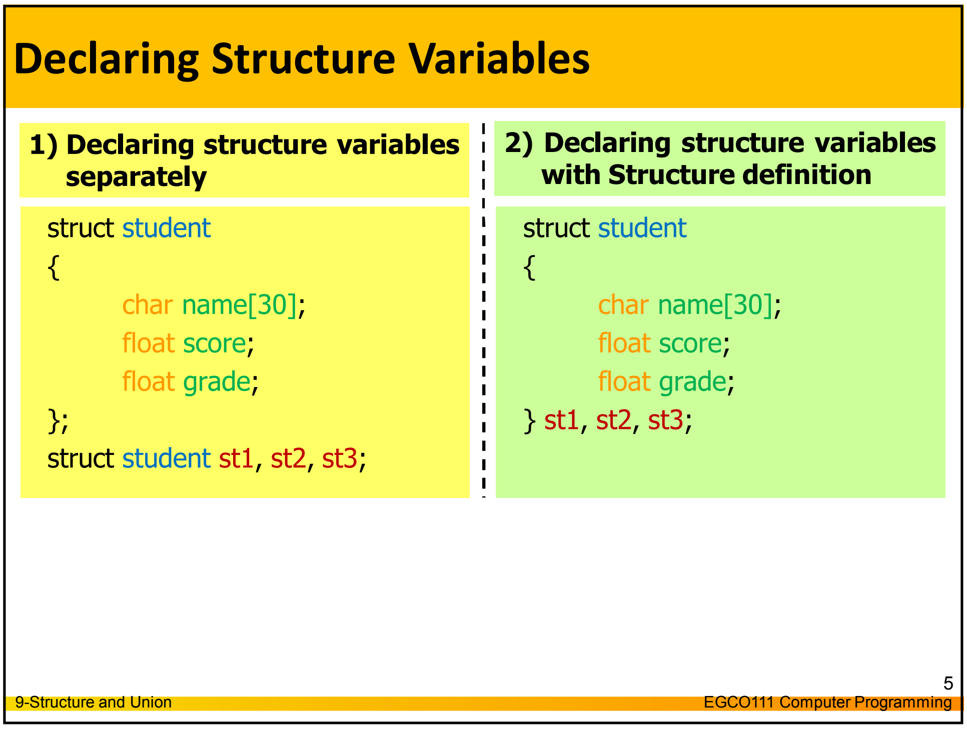
Declaring Structure Variables

1) Declaring structure variables separately

```
struct struct_name
{
        type name_1 ;
        type name_2 ;
        ...........
        type name_n ;
};
struct struct_name svar_1, svar_2,
… , svar_m;
```

2) Declaring structure variables with Structure definition

```
struct struct_name
{
        type name_1 ;
        type name_2 ;
        ...........
        type name_n ;
} svar_1, svar_2, … , svar_m;
```



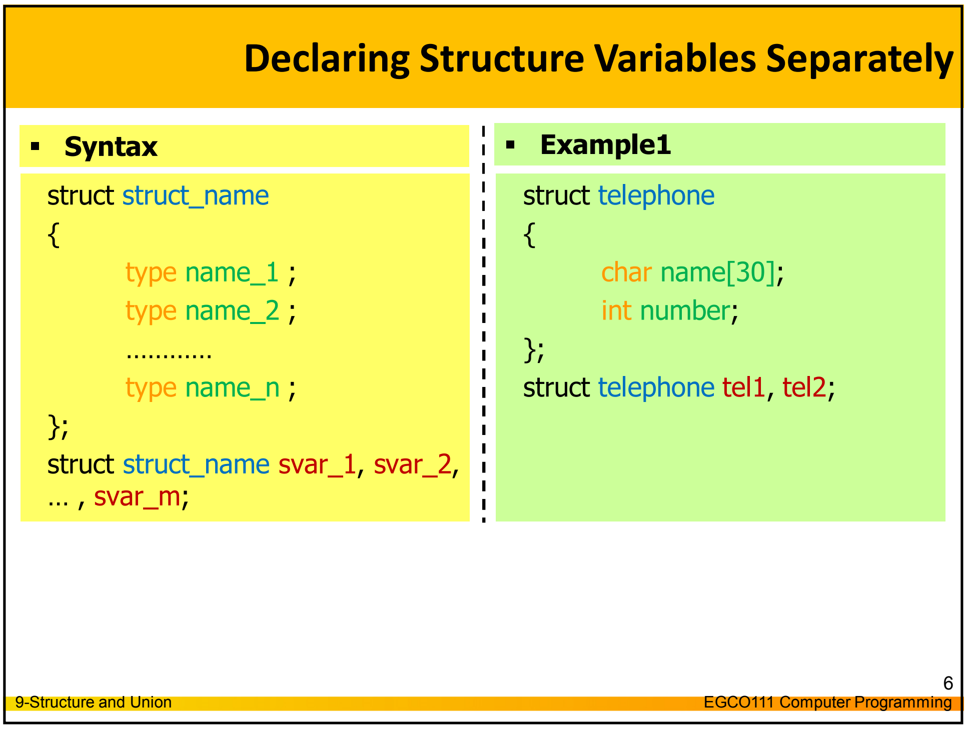
Declaring Structure Variables

1) Declaring structure variables separately

```
struct student
{
        char name[30];
        float score;
        float grade;
};
struct student st1, st2, st3;
```

2) Declaring structure variables with Structure definition

```
struct student
{
        char name[30];
        float score;
        float grade;
} st1, st2, st3;
```



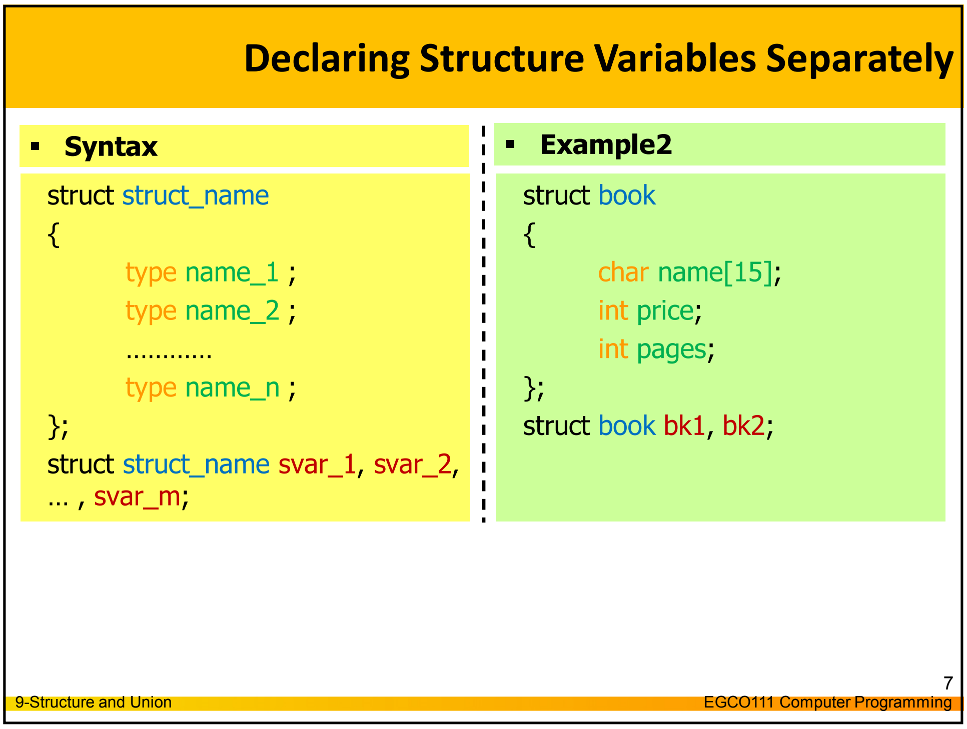
Declaring Structure Variables Separately

- **Syntax**

```
struct struct_name
{
        type name_1 ;
        type name_2 ;
        …………
        type name_n ;
};
struct struct_name svar_1, svar_2,
… , svar_m;
```

- **Example1**

```
struct telephone
{
        char name[30];
        int number;
};
struct telephone tel1, tel2;
```



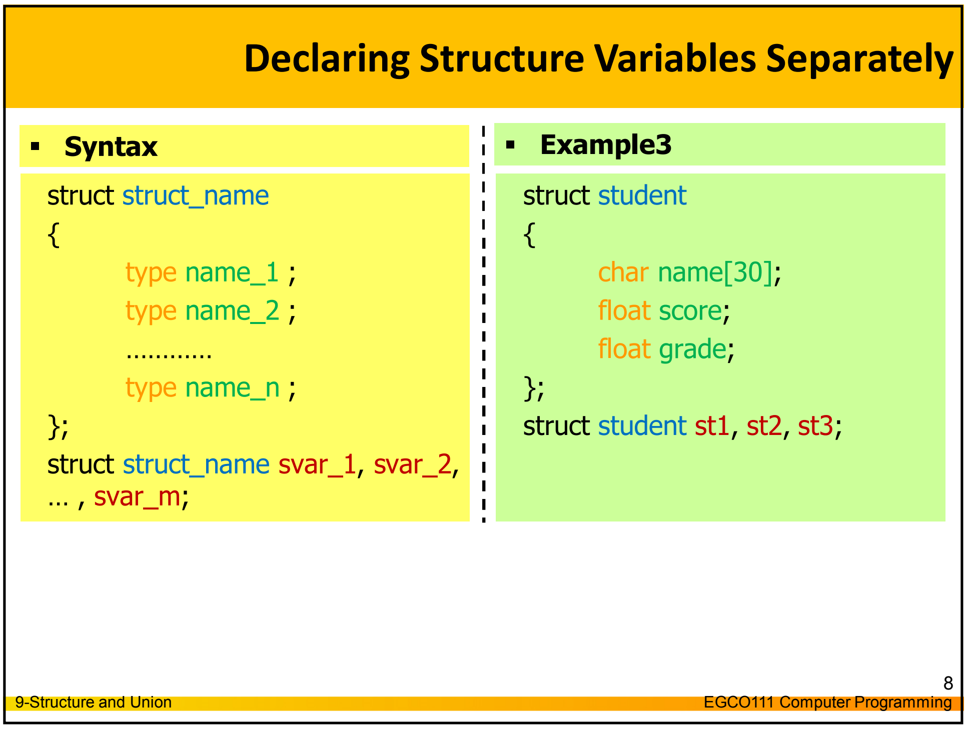
Declaring Structure Variables Separately

- **Syntax**

```
struct struct_name
{
        type name_1 ;
        type name_2 ;
        …………
        type name_n ;
};
struct struct_name svar_1, svar_2,
… , svar_m;
```

- **Example2**

```
struct book
{
        char name[15];
        int price;
        int pages;
};
struct book bk1, bk2;
```



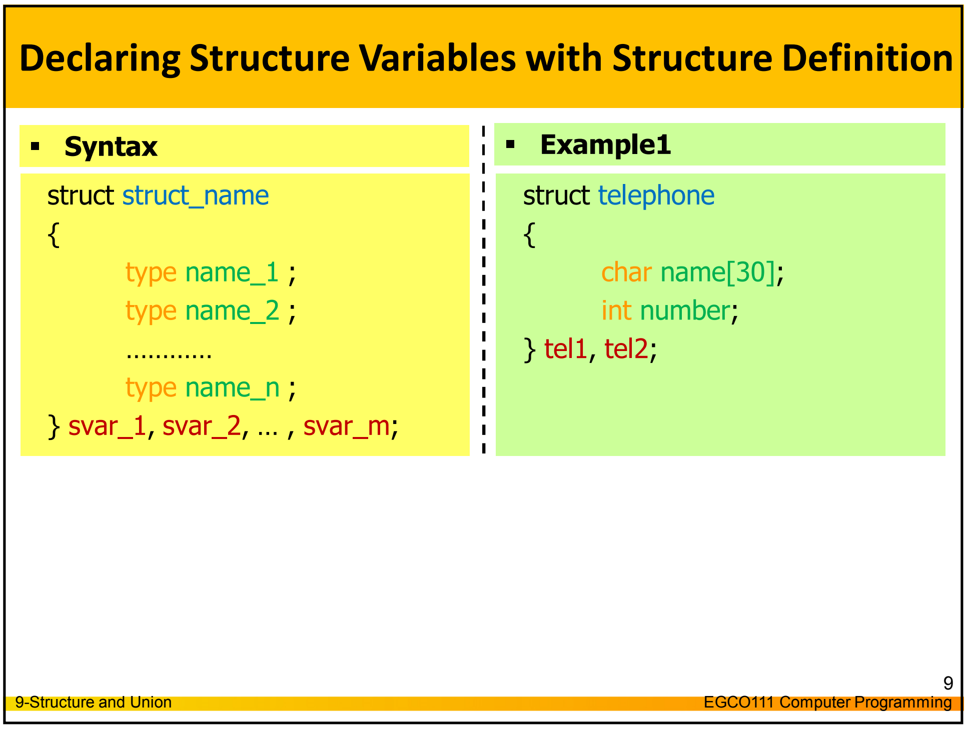
Declaring Structure Variables Separately

- **Syntax**

```
struct struct_name
{
        type name_1 ;
        type name_2 ;
        …………
        type name_n ;
};
struct struct_name svar_1, svar_2,
… , svar_m;
```

- **Example3**

```
struct student
{
        char name[30];
        float score;
        float grade;
};
struct student st1, st2, st3;
```



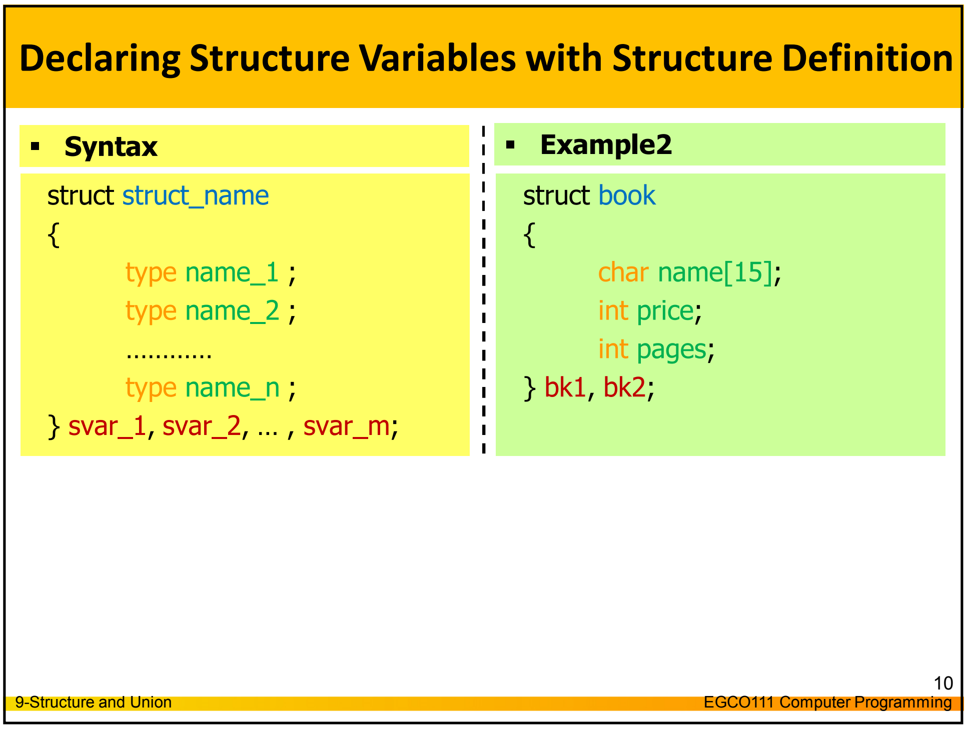
Declaring Structure Variables with Structure Definition

- **Syntax**

```
struct struct_name
{
        type name_1 ;
        type name_2 ;
        …………
        type name_n ;
} svar_1, svar_2, … , svar_m;
```

- **Example1**

```
struct telephone
{
        char name[30];
        int number;
} tel1, tel2;
```



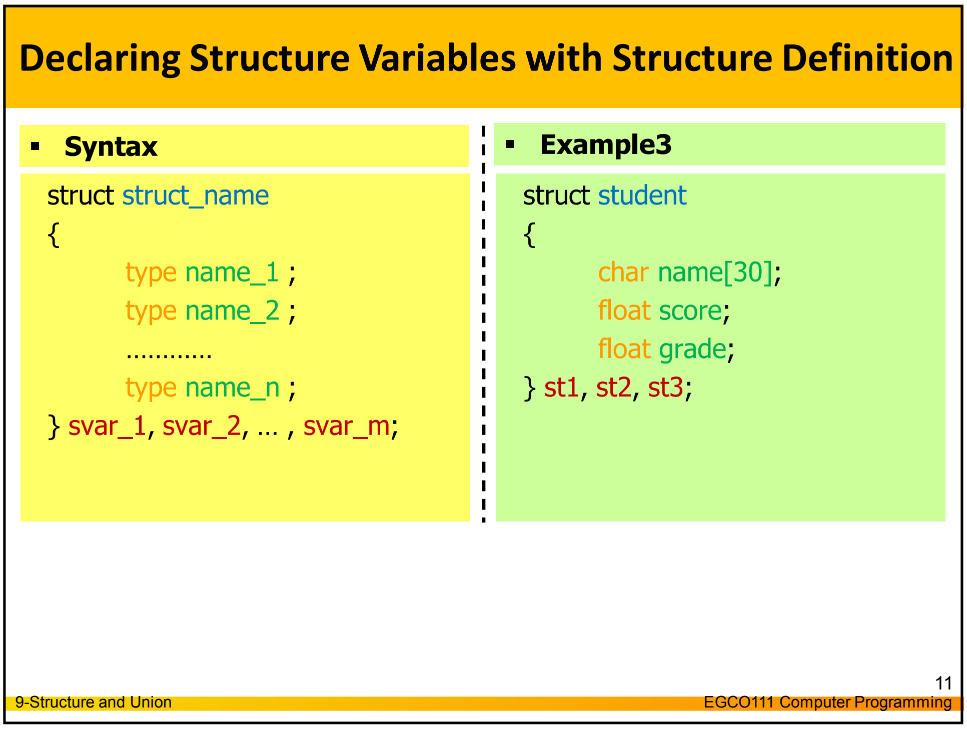
Declaring Structure Variables with Structure Definition

- **Syntax**

```
struct struct_name
{
        type name_1 ;
        type name_2 ;
        …………
        type name_n ;
} svar_1, svar_2, … , svar_m;
```

- **Example2**

```
struct book
{
        char name[15];
        int price;
        int pages;
} bk1, bk2;
```



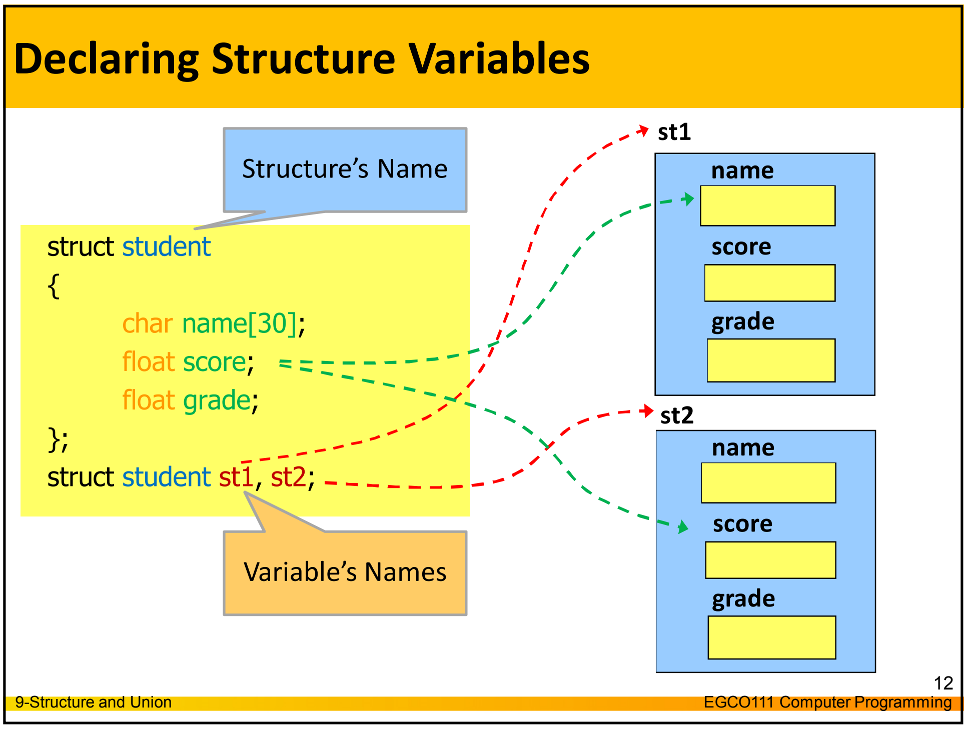
Declaring Structure Variables with Structure Definition

- **Syntax**

```
struct struct_name
{
        type name_1 ;
        type name_2 ;
        …………
        type name_n ;
} svar_1, svar_2, … , svar_m;
```

- **Example3**

```
struct student
{
        char name[30];
        float score;
        float grade;
} st1, st2, st3;
```



Declaring Structure Variables

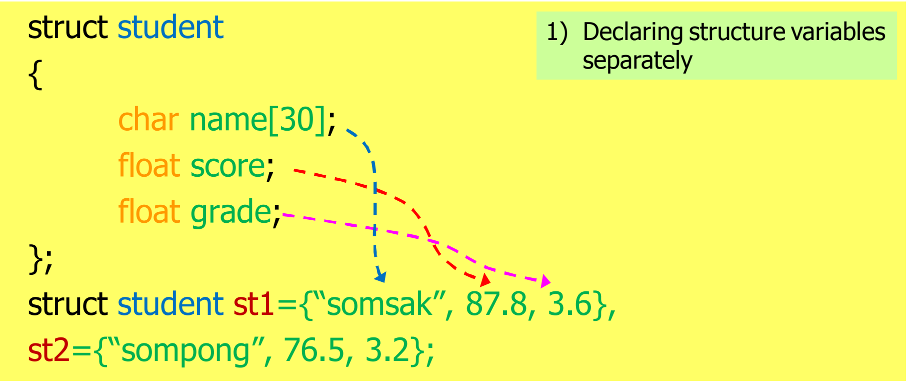
Structure's Name

```
struct student
{
        char name[30];
        float score;
        float grade;
};
struct student st1, st2;
```

Variable's Names

Structure Initialization

- Like any other data type, structure variable can also be initialized at compile time.

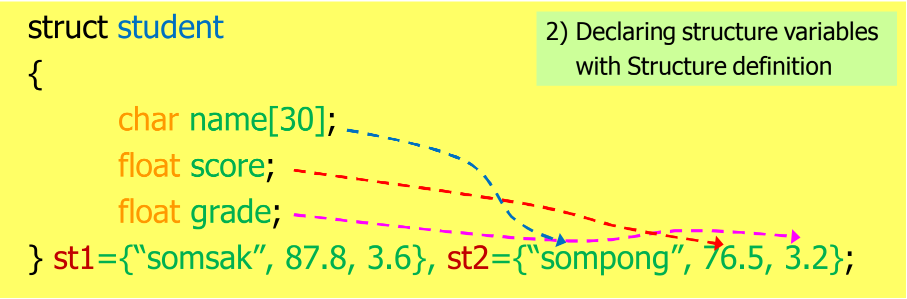


1) Declaring structure variables separately

```
struct student
{
        char name[30];
        float score;
        float grade;
};
struct student st1={"somsak", 87.8, 3.6},
st2={"sompong", 76.5, 3.2};
```

Structure Initialization

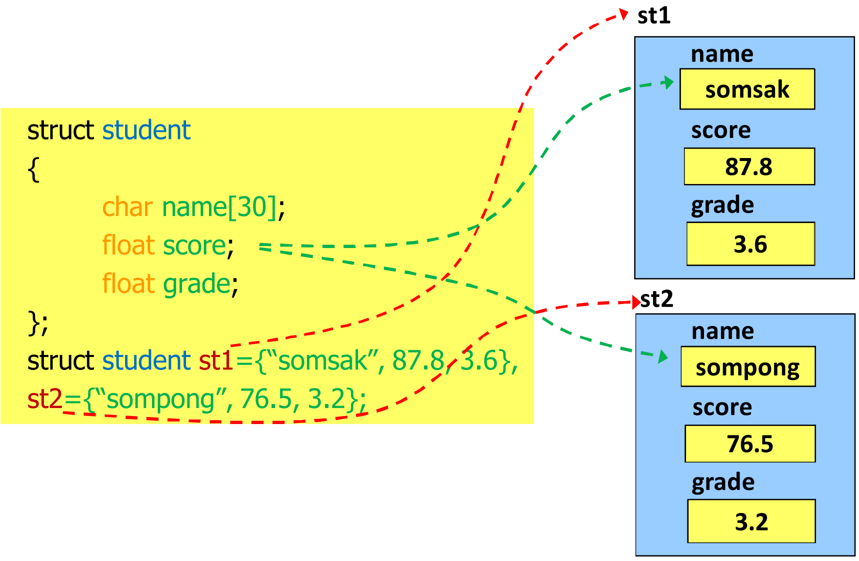
- Like any other data type, structure variable can also be initialized at compile time.



2) Declaring structure variables with Structure definition

```
struct student
{
        char name[30];
        float score;
        float grade;
} st1={"somsak", 87.8, 3.6}, st2={"sompong", 76.5, 3.2};
```

Structure Initialization



```
struct student
{
        char name[30];
        float score;
        float grade;
};
struct student st1={"somsak", 87.8, 3.6},
st2={"sompong", 76.5, 3.2};
```

Accessing Structure Members

- Structure members can be accessed and assigned values in number of ways.
- In order to assign value, the member must be linked with the structure variable using (.) operator.

- For example:

```
st1.name = "somsak";
st1.score = 77.8;
printf("%s\n", st1.name);
printf("%.2f\n", st1.score);
```

```
struct student {
        char name[30];
        float score;
        float grade;
};
struct student st1;
```

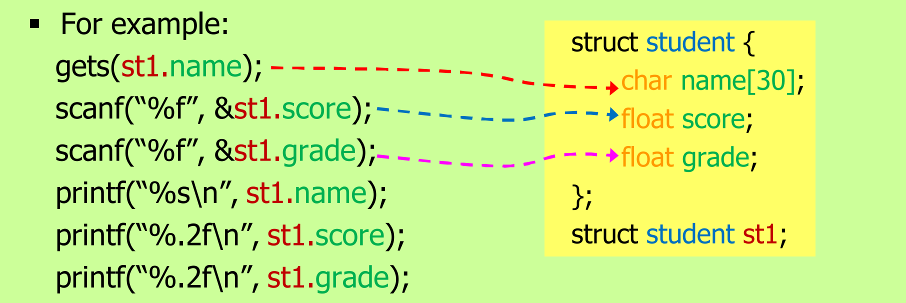
- Result:

```
somsak
77.8
```

Accessing Structure Members

- We can use gets() or scanf() to give values to structure members through keyboard.

- For example:



```
gets(st1.name);
scanf("%f", &st1.score);
scanf("%f", &st1.grade);
printf("%s\n", st1.name);
printf("%.2f\n", st1.score);
printf("%.2f\n", st1.grade);
```

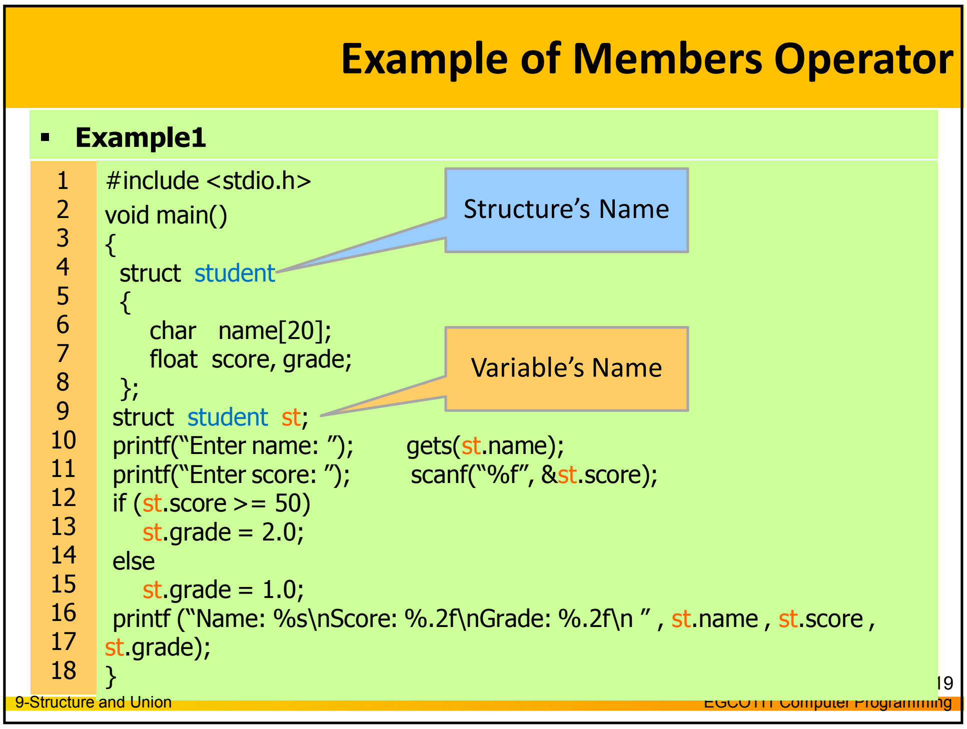
```
struct student {
char name[30];
float score;
float grade;
};
struct student st1;
```

- Result:

```
somsak
77.8
3.6
```

Members Operator

- Member operator (.) is used for accessing data.
- Structure variables are similar to the other basic variables.
- Basic operations, such as
  - assign operation,
  - logical operation, and
  - mathematical operation, can be applied to them.



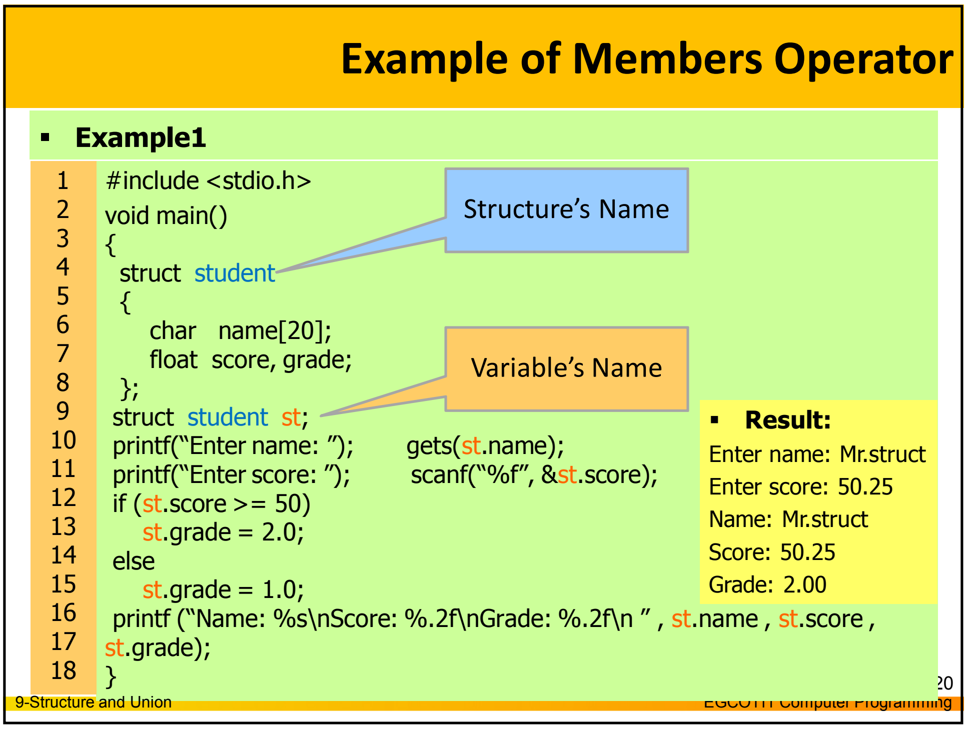
Example of Members Operator

- **Example1**

```
#include <stdio.h>
void main()
{
  struct  student
  {
      char   name[20];
      float  score, grade;
  };
 struct  student  st;
 printf("Enter name: ");        gets(st.name);
 printf("Enter score: ");        scanf("%f", &st.score);
 if (st.score >= 50)
     st.grade = 2.0;
 else
     st.grade = 1.0;
 printf ("Name: %s\nScore: %.2f\nGrade: %.2f\n " , st.name , st.score ,
st.grade);
}
```

Structure's Name

Variable's Name



Example of Members Operator

- **Example1**

```
#include <stdio.h>
void main()
{
  struct  student
  {
      char   name[20];
      float  score, grade;
  };
 struct  student  st;
 printf("Enter name: ");        gets(st.name);
 printf("Enter score: ");        scanf("%f", &st.score);
 if (st.score >= 50)
     st.grade = 2.0;
 else
     st.grade = 1.0;
 printf ("Name: %s\nScore: %.2f\nGrade: %.2f\n " , st.name , st.score ,
st.grade);
}
```

Structure's Name

Variable's Name

- **Result:**

```
Enter name: Mr.struct
Enter score: 50.25
Name: Mr.struct
Score: 50.25
Grade: 2.00
```

Question ?

- **Question1**

```
#include <stdio.h>
void main()
{
   struct  student
   {
       char   name[20];
       float   score, grade;
   };
   struct  student  st1={"somsak", 87.8, 3.6}, st2;
   st2.score=76.5;
   if (st1 > st2)
      printf("you are great");
}
```

Exercise1

- **Write a program to get information name, surname, address, and salary.**
- **Example of result:**

```
Enter name: Mr.struct
Enter surname: members
Enter address: mahidol University
Enter salary: 35000

Mr.struct members
mahidol University
35000.00
Your salary is greater than 25000
```

```
Enter name: Mr.struct
Enter surname: members
Enter address: mahidol University
Enter salary: 15000

Mr.struct members
mahidol University
15000.00
Your salary is lower than 25000
```

Operations on Structure Variables

```
struct  student
{
    char   name[20];
    float  score, grade;
} st1, st2;
```

- We can use
```
if (st1.score >= 80)
   st1.grade = 4.0;
```

- We cannot use
```
if (st2 > st1)
   printf(“you are great”);
```

- We can use
```
if (st2.score > st1.score)
   printf(“you are great”);
```

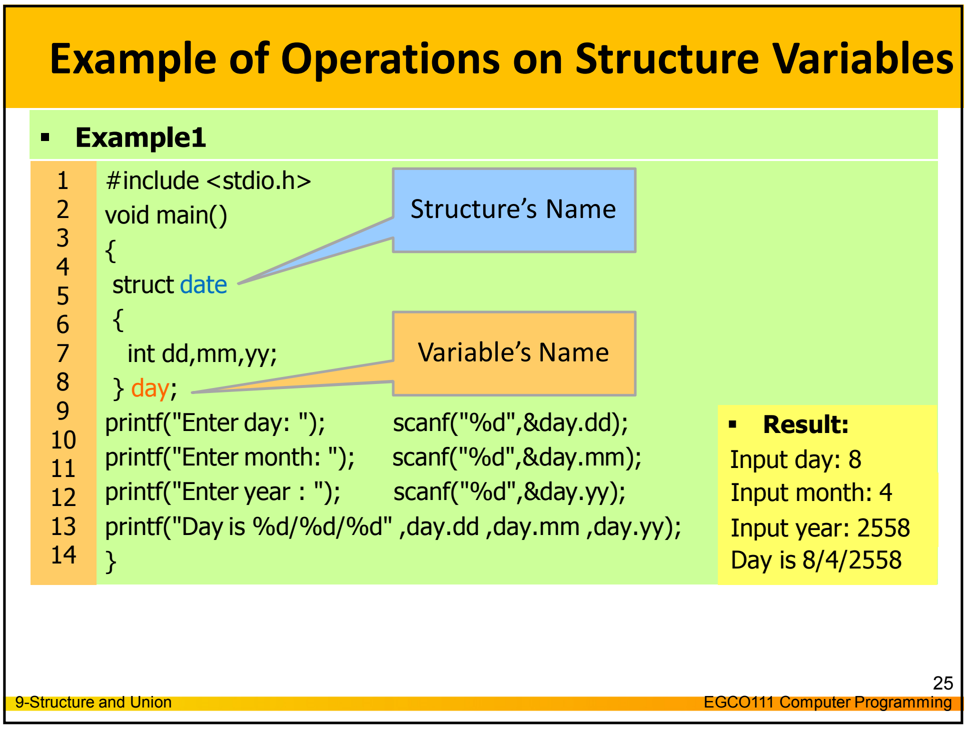
Example of Operations on Structure Variables

- **Example1**

```
#include <stdio.h>
void main()
{
 struct date
 {
   int dd,mm,yy;
 } day;
printf("Enter day: ");        scanf("%d",&day.dd);
printf("Enter month: ");      scanf("%d",&day.mm);
printf("Enter year : ");      scanf("%d",&day.yy);
printf("Day is %d/%d/%d" ,day.dd ,day.mm ,day.yy);
}
```

Structure’s Name

Variable’s Name



Example of Operations on Structure Variables

- **Example1**

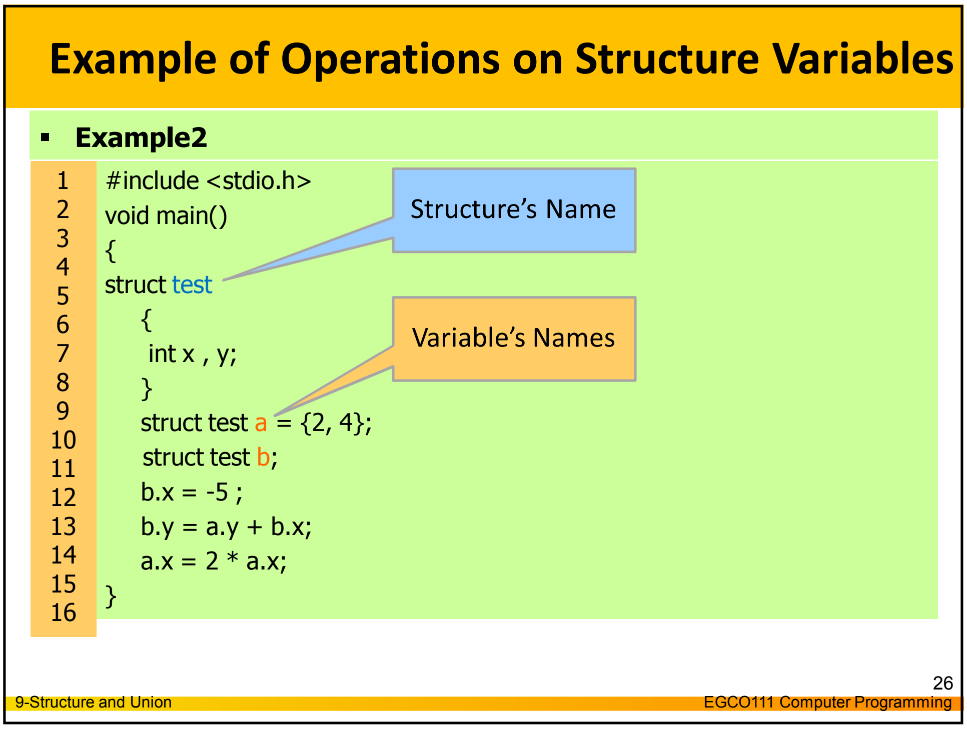
```
#include <stdio.h>
void main()
{
 struct date
 {
   int dd,mm,yy;
 } day;
printf("Enter day: ");        scanf("%d",&day.dd);
printf("Enter month: ");      scanf("%d",&day.mm);
printf("Enter year : ");      scanf("%d",&day.yy);
printf("Day is %d/%d/%d" ,day.dd ,day.mm ,day.yy);
}
```

Structure's Name

Variable's Name

- **Result:**

```
Input day: 8
Input month: 4
Input year: 2558
Day is 8/4/2558
```



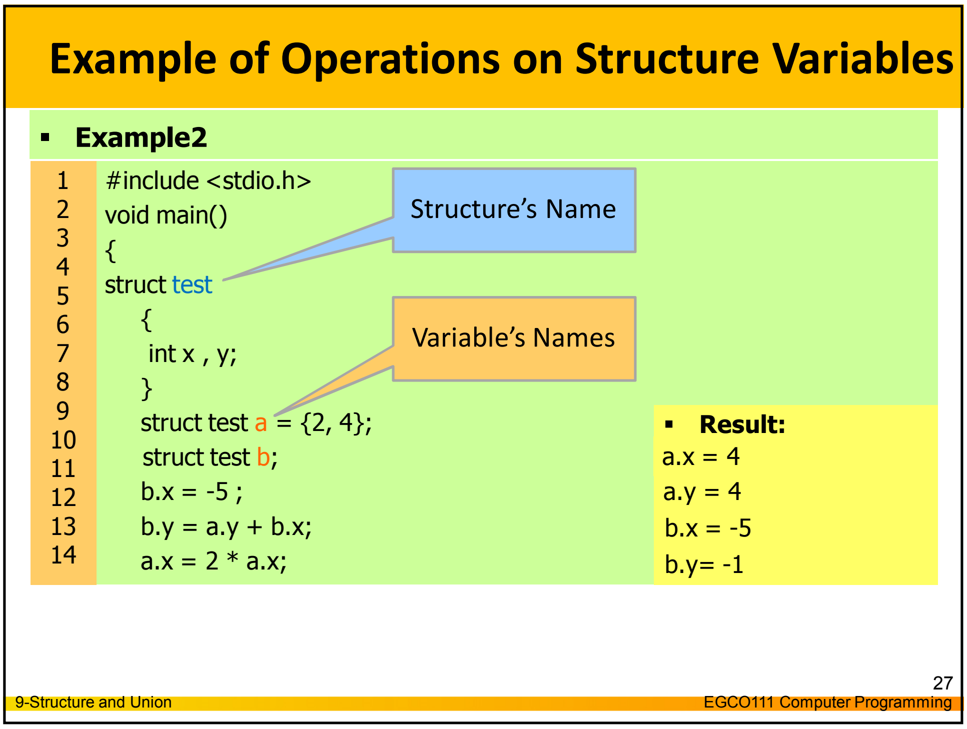
Example of Operations on Structure Variables

- **Example2**

```
#include <stdio.h>
void main()
{
struct test
     {
      int x , y;
     }
     struct test a = {2, 4};
     struct test b;
     b.x = -5 ;
     b.y = a.y + b.x;
     a.x = 2 * a.x;
}
```

Structure's Name

Variable's Names



Example of Operations on Structure Variables

- **Example2**

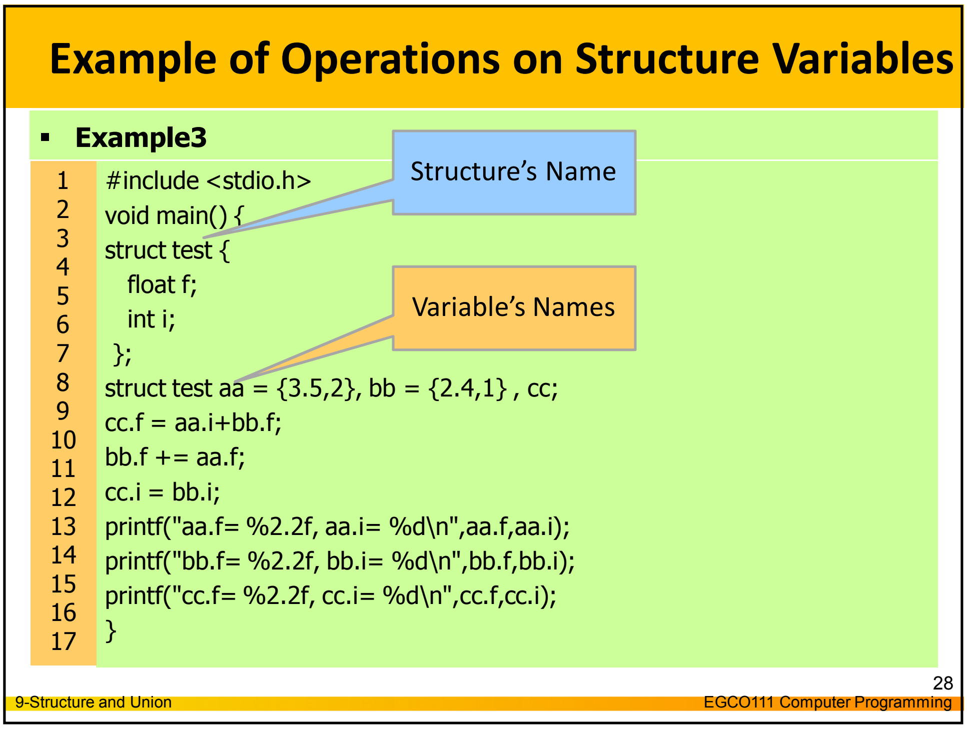
```
#include <stdio.h>
void main()
{
struct test
    {
     int x , y;
    }
    struct test a = {2, 4};
    struct test b;
    b.x = -5 ;
    b.y = a.y + b.x;
    a.x = 2 * a.x;
```

Structure’s Name

Variable’s Names

- **Result:**

```
a.x = 4
a.y = 4
b.x = -5
b.y= -1
```



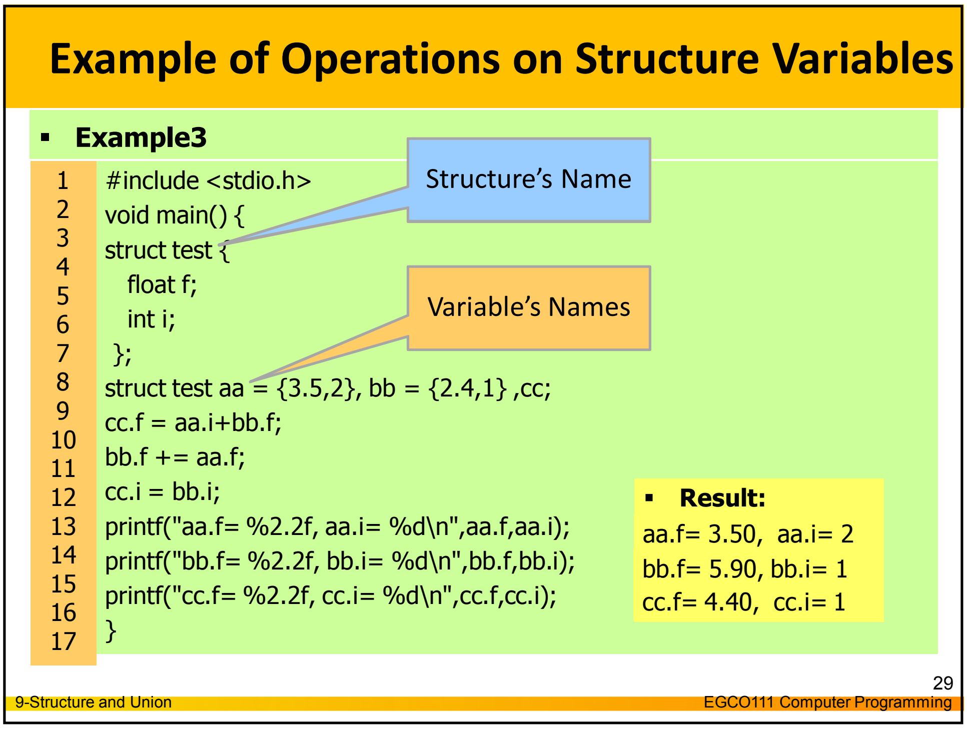
Example of Operations on Structure Variables

- **Example3**

```
#include <stdio.h>
void main() {
struct test {
   float f;
   int i;
 };
struct test aa = {3.5,2}, bb = {2.4,1} , cc;
cc.f = aa.i+bb.f;
bb.f += aa.f;
cc.i = bb.i;
printf("aa.f= %2.2f, aa.i= %d\n",aa.f,aa.i);
printf("bb.f= %2.2f, bb.i= %d\n",bb.f,bb.i);
printf("cc.f= %2.2f, cc.i= %d\n",cc.f,cc.i);
}
```

Structure’s Name

Variable’s Names



Example of Operations on Structure Variables

- **Example3**

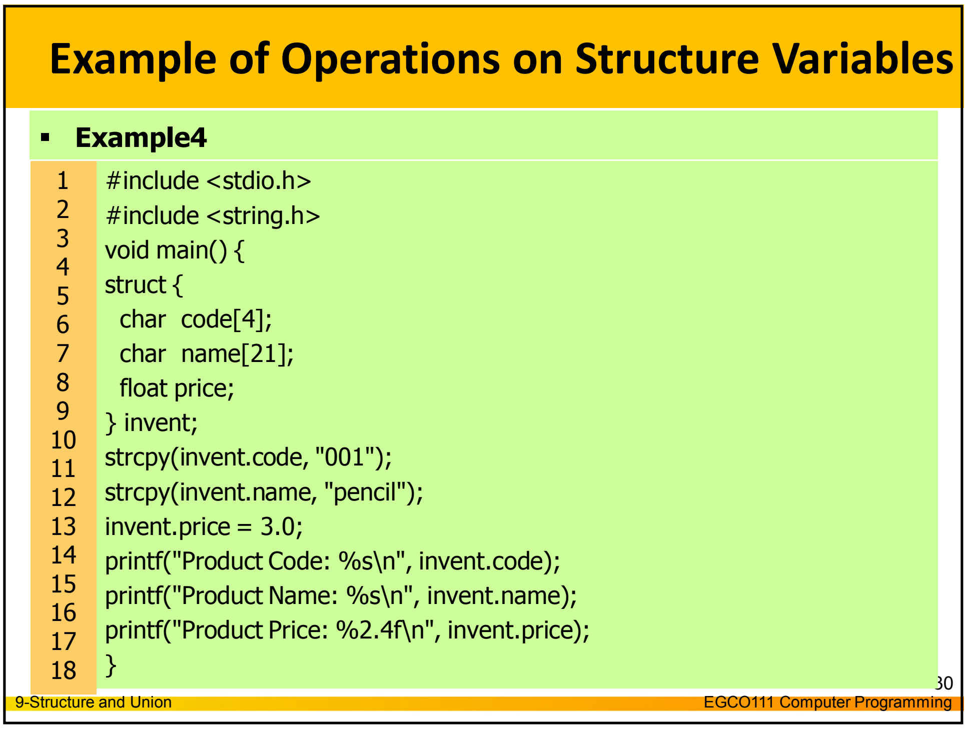
```
#include <stdio.h>
void main() {
struct test {
   float f;
   int i;
 };
struct test aa = {3.5,2}, bb = {2.4,1} ,cc;
cc.f = aa.i+bb.f;
bb.f += aa.f;
cc.i = bb.i;
printf("aa.f= %2.2f, aa.i= %d\n",aa.f,aa.i);
printf("bb.f= %2.2f, bb.i= %d\n",bb.f,bb.i);
printf("cc.f= %2.2f, cc.i= %d\n",cc.f,cc.i);
}
```

Structure's Name

Variable's Names

- **Result:**

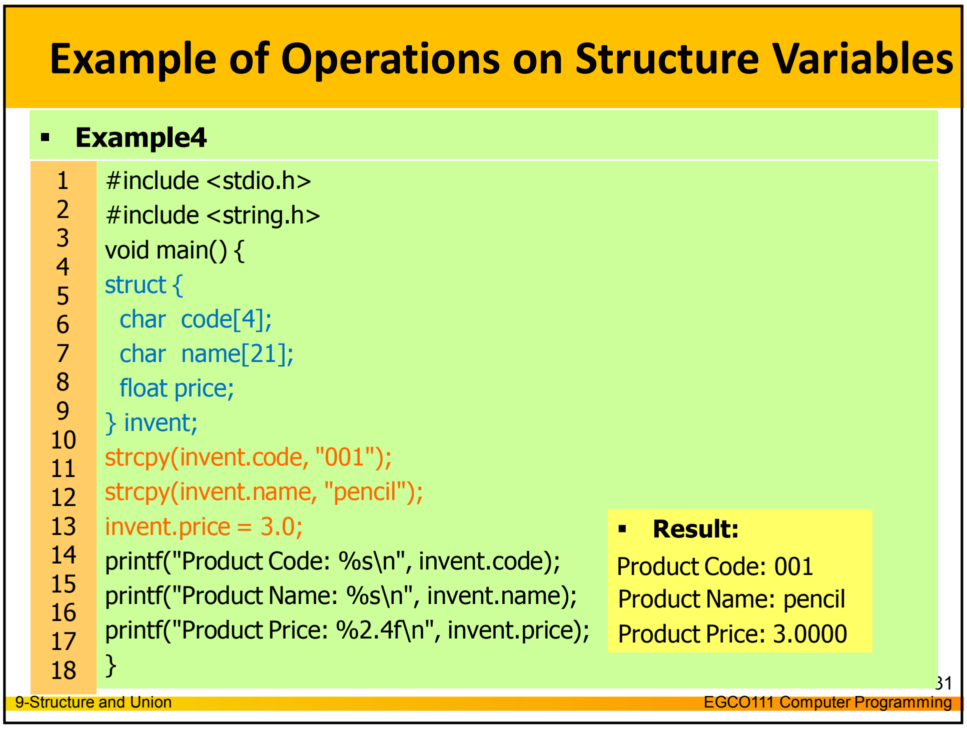
```
aa.f= 3.50,  aa.i= 2
bb.f= 5.90, bb.i= 1
cc.f= 4.40,  cc.i= 1
```



Example of Operations on Structure Variables

- **Example4**

```
#include <stdio.h>
#include <string.h>
void main() {
struct {
  char  code[4];
  char  name[21];
  float price;
} invent;
strcpy(invent.code, "001");
strcpy(invent.name, "pencil");
invent.price = 3.0;
printf("Product Code: %s\n", invent.code);
printf("Product Name: %s\n", invent.name);
printf("Product Price: %2.4f\n", invent.price);
}
```



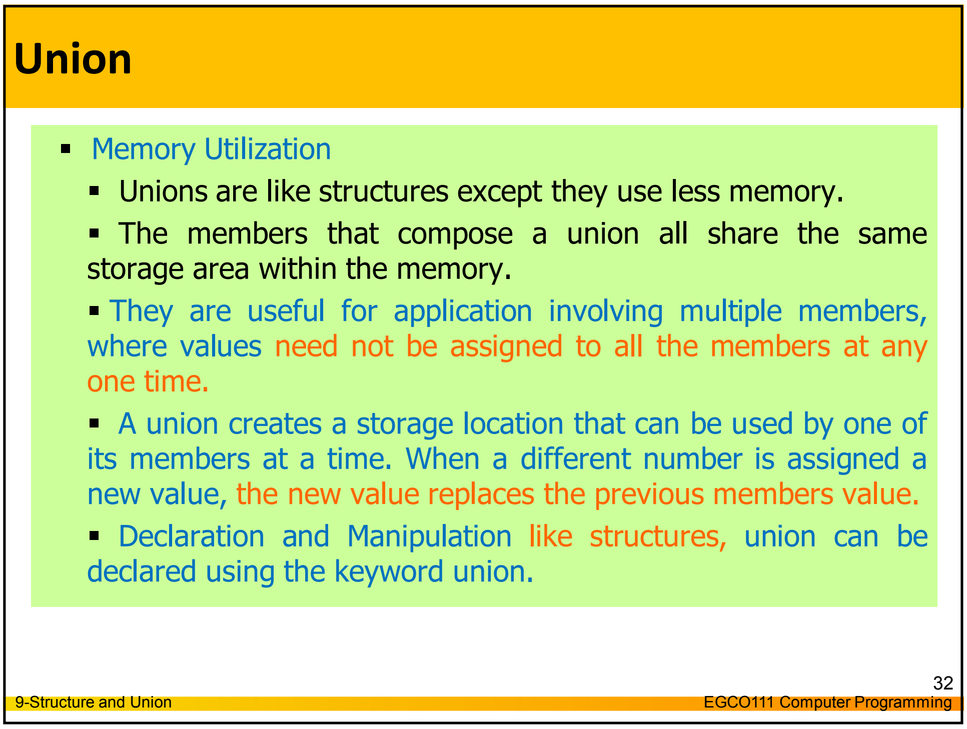
Example of Operations on Structure Variables

- **Example4**

```
#include <stdio.h>
#include <string.h>
void main() {
struct {
  char  code[4];
  char  name[21];
  float price;
} invent;
strcpy(invent.code, "001");
strcpy(invent.name, "pencil");
invent.price = 3.0;
printf("Product Code: %s\n", invent.code);
printf("Product Name: %s\n", invent.name);
printf("Product Price: %2.4f\n", invent.price);
}
```

- **Result:**

```
Product Code: 001
Product Name: pencil
Product Price: 3.0000
```



Union

- Memory Utilization
  - Unions are like structures except they use less memory.
  - The members that compose a union all share the same storage area within the memory.
  - They are useful for application involving multiple members, where values need not be assigned to all the members at any one time.
  - A union creates a storage location that can be used by one of its members at a time. When a different number is assigned a new value, the new value replaces the previous members value.
  - Declaration and Manipulation like structures, union can be declared using the keyword union.

Union

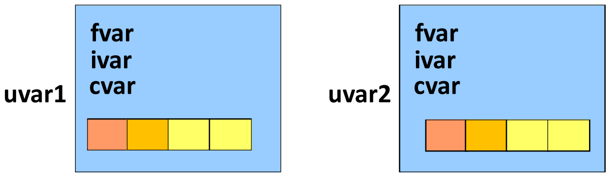
```
union union_name
{
        type name_1 ;
        type name_2 ;
        …………
        type name_n ;
};
union union_name uvar_1, uvar_2, … , uvar_m;
```

Example of Union

```
union uvar
{
   float fvar;
   int ivar;
   char cvar;
} uvar1, uvar2;
```

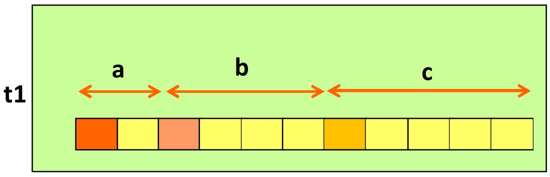
Union's Name

Variable's Names

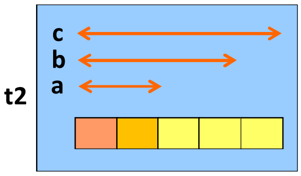


Structure VS. Union

```
struct  test1
{
   int a;
   float b;
   char c[5];
} t1;
```



```
union test2
{
   int a;
   float b;
   char c[5];
} t2;
```



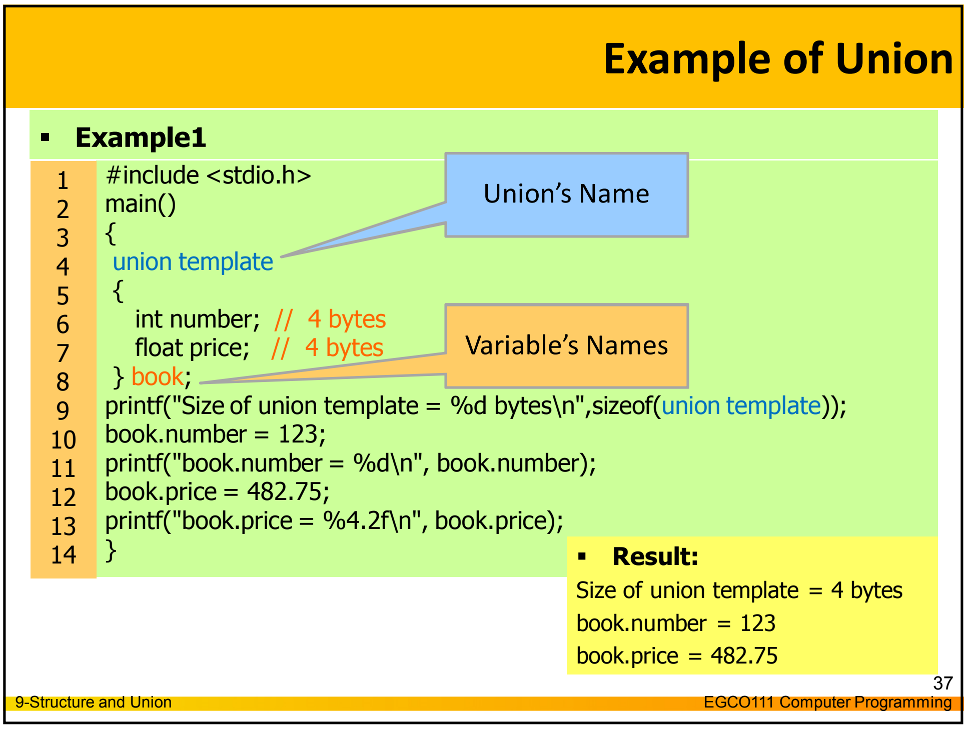
Example of Union

- **Example1**

```
#include <stdio.h>
main()
{
 union template
 {
    int number;
    float price;
 } book;
printf("Size of union template = %d bytes\n",sizeof(union template));
book.number = 123;
printf("book.number = %d\n", book.number);
book.price = 482.75;
printf("book.price = %4.2f\n", book.price);
}
```

Union’s Name

Variable’s Names



Example of Union

- **Example1**

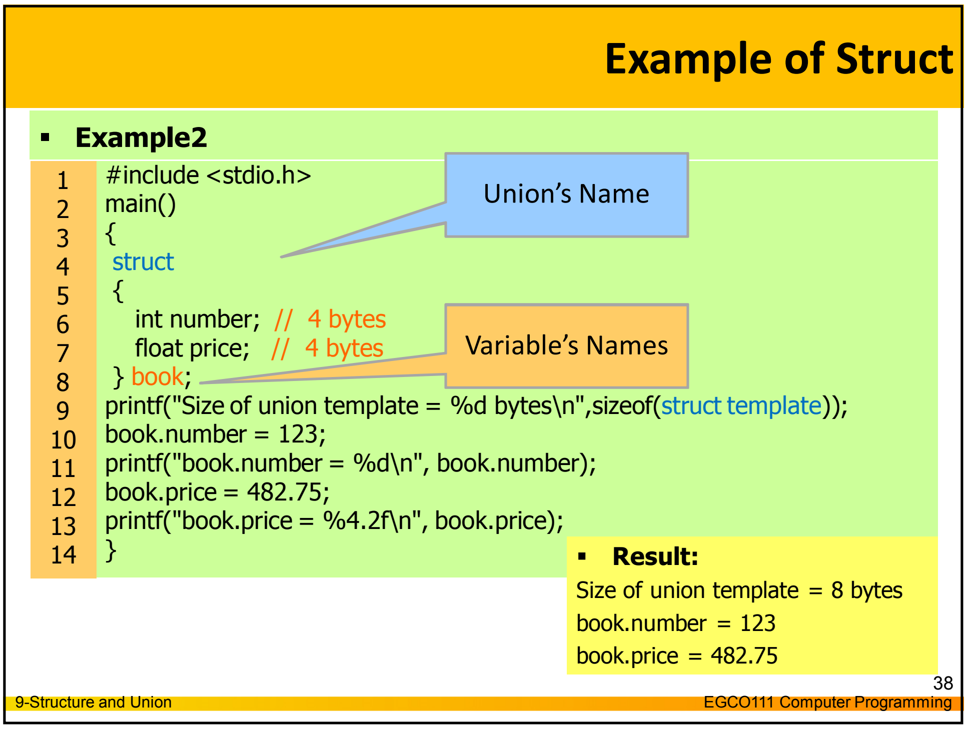
```
#include <stdio.h>
main()
{
 union template
 {
    int number;  //  4 bytes
    float price;   //  4 bytes
 } book;
printf("Size of union template = %d bytes\n",sizeof(union template));
book.number = 123;
printf("book.number = %d\n", book.number);
book.price = 482.75;
printf("book.price = %4.2f\n", book.price);
}
```

Union's Name

Variable's Names

- **Result:**

```
Size of union template = 4 bytes
book.number = 123
book.price = 482.75
```



Example of Struct

- **Example2**

```
#include <stdio.h>
main()
{
 struct
 {
    int number;  //  4 bytes
    float price;   //  4 bytes
 } book;
printf("Size of union template = %d bytes\n",sizeof(struct template));
book.number = 123;
printf("book.number = %d\n", book.number);
book.price = 482.75;
printf("book.price = %4.2f\n", book.price);
}
```

Union's Name

Variable's Names

- **Result:**

```
Size of union template = 8 bytes
book.number = 123
book.price = 482.75
```

Array of Structure

- **Sytax1**

```
struct  student {
    char      name[30];
    double  score, grade;
} st[40]; //Declaration
```

- **Sytax2**

```
struct  student
{
    char      name[30];
    double  score, grade;
};
struct  student st[40]; //Declaration
```

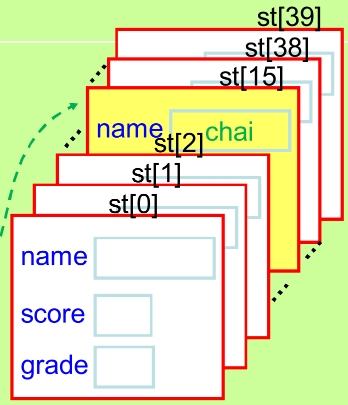
- Manipulation: *variable_name*[index].member

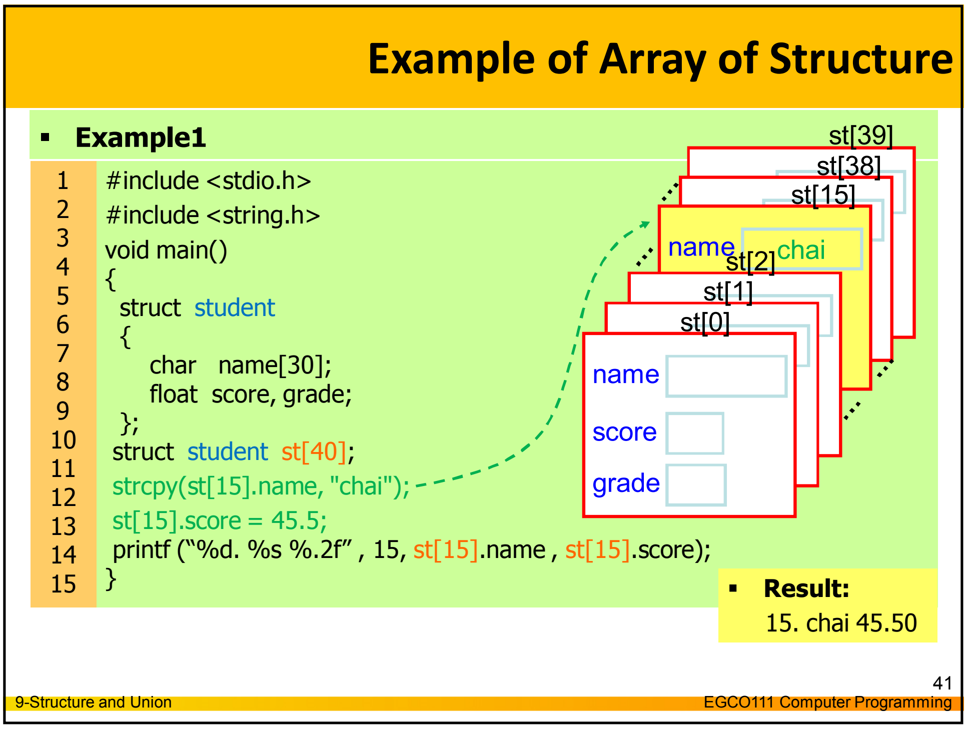
```
strcpy(st[15].name, "chai");
st[15].score = 45.5;
```

Example of Array of Structure

- **Example1**

```
#include <stdio.h>
#include <string.h>
void main()
{
  struct  student
  {
      char   name[30];
      float  score, grade;
  };
 struct  student  st[40];
 strcpy(st[15].name, "chai");
 st[15].score = 45.5;
 printf (“%d. %s %.2f” , 15, st[15].name , st[15].score);
}
```





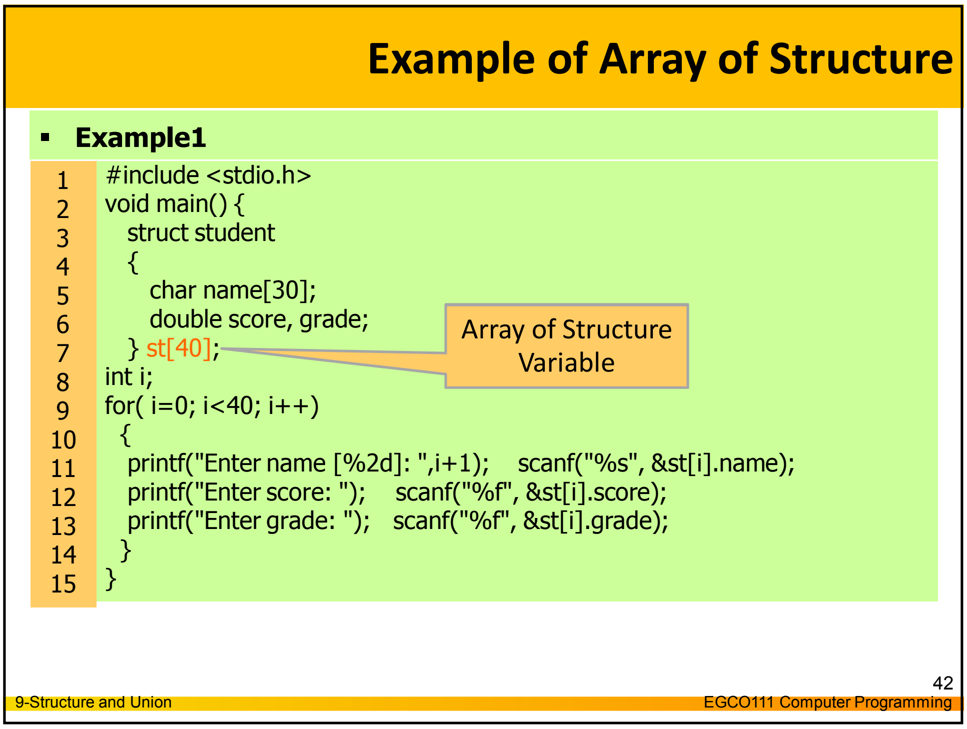
Example of Array of Structure

- **Example1**

```
#include <stdio.h>
#include <string.h>
void main()
{
  struct  student
  {
      char   name[30];
      float  score, grade;
  };
 struct  student  st[40];
 strcpy(st[15].name, "chai");
 st[15].score = 45.5;
 printf (“%d. %s %.2f” , 15, st[15].name , st[15].score);
}
```

st[39]
st[38]
st[15]
name
chai
st[2]
st[1]
st[0]
name
score
grade

- **Result:**
  15. chai 45.50

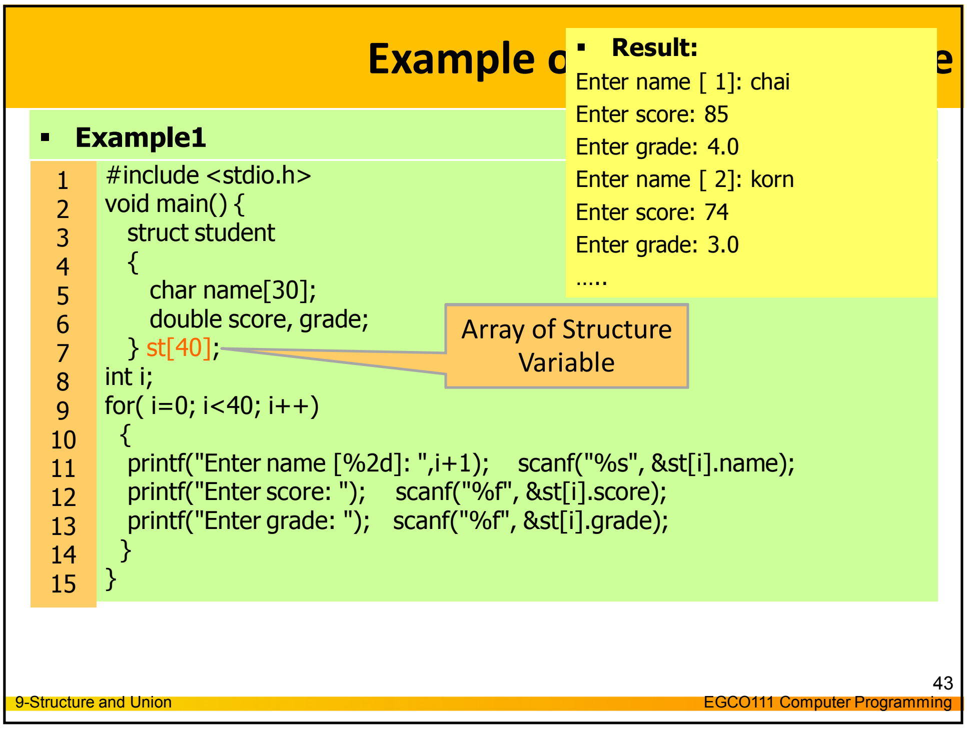


Example of Array of Structure

- **Example1**

```
#include <stdio.h>
void main() {
   struct student
   {
      char name[30];
      double score, grade;
   } st[40];
int i;
for( i=0; i<40; i++)
  {
   printf("Enter name [%2d]: ",i+1);    scanf("%s", &st[i].name);
   printf("Enter score: ");    scanf("%f", &st[i].score);
   printf("Enter grade: ");    scanf("%f", &st[i].grade);
  }
}
```

Array of Structure Variable



Example o e

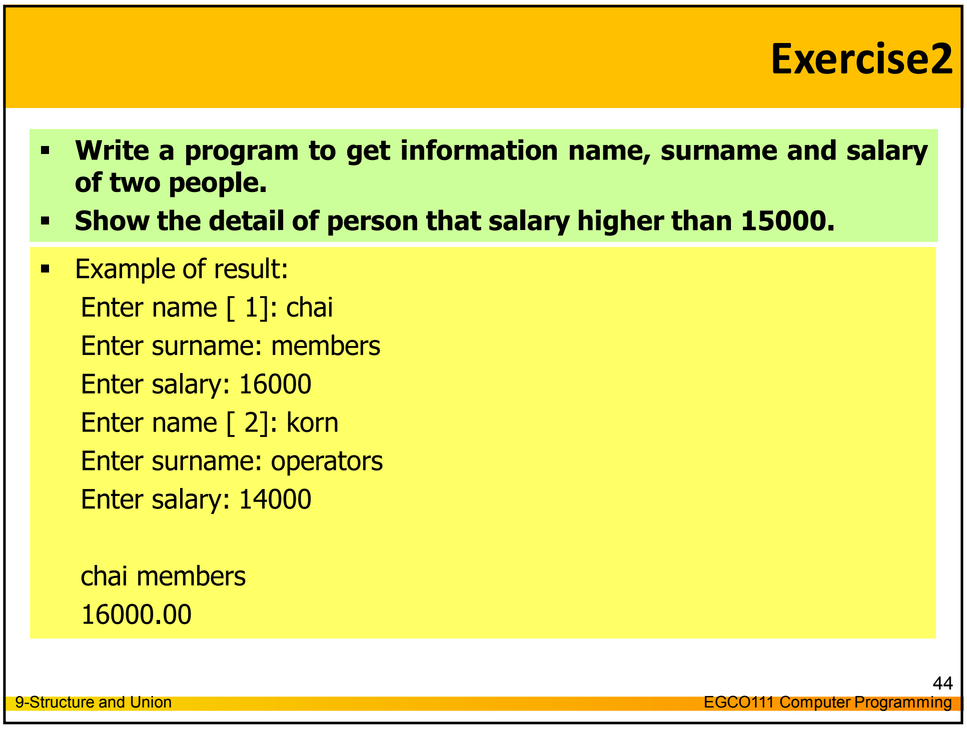
- **Example1**

```
#include <stdio.h>
void main() {
   struct student
   {
      char name[30];
      double score, grade;
   } st[40];
int i;
for( i=0; i<40; i++)
  {
   printf("Enter name [%2d]: ",i+1);    scanf("%s", &st[i].name);
   printf("Enter score: ");    scanf("%f", &st[i].score);
   printf("Enter grade: ");    scanf("%f", &st[i].grade);
  }
}
```

- **Result:**

```
Enter name [ 1]: chai
Enter score: 85
Enter grade: 4.0
Enter name [ 2]: korn
Enter score: 74
Enter grade: 3.0
.....
```

Array of Structure Variable



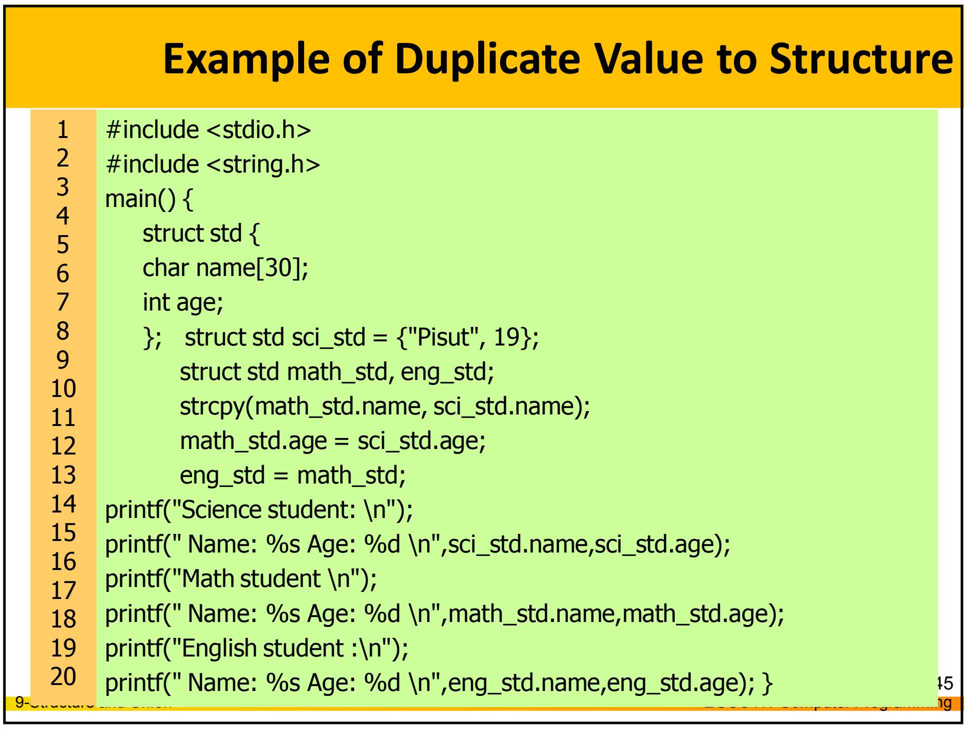
Exercise2

- **Write a program to get information name, surname and salary of two people.**
- **Show the detail of person that salary higher than 15000.**

- Example of result:

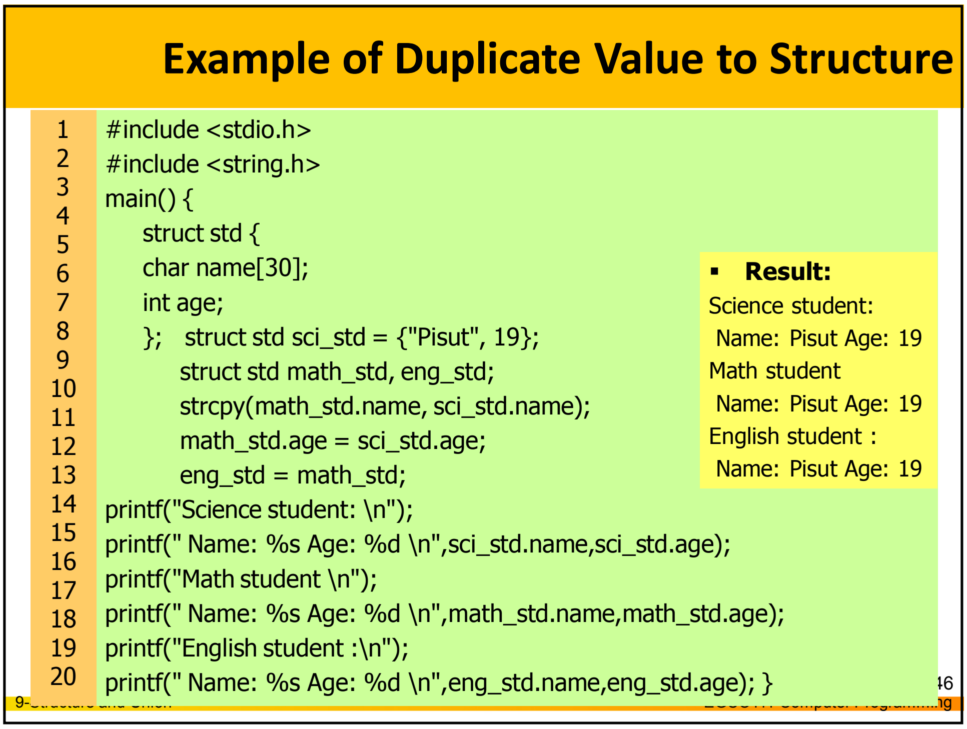
```
Enter name [ 1]: chai
Enter surname: members
Enter salary: 16000
Enter name [ 2]: korn
Enter surname: operators
Enter salary: 14000

chai members
16000.00
```



Example of Duplicate Value to Structure

```
#include <stdio.h>
#include <string.h>
main() {
     struct std {
     char name[30];
     int age;
     };   struct std sci_std = {"Pisut", 19};
          struct std math_std, eng_std;
          strcpy(math_std.name, sci_std.name);
          math_std.age = sci_std.age;
          eng_std = math_std;
printf("Science student: \n");
printf(" Name: %s Age: %d \n",sci_std.name,sci_std.age);
printf("Math student \n");
printf(" Name: %s Age: %d \n",math_std.name,math_std.age);
printf("English student :\n");
printf(" Name: %s Age: %d \n",eng_std.name,eng_std.age); }
```

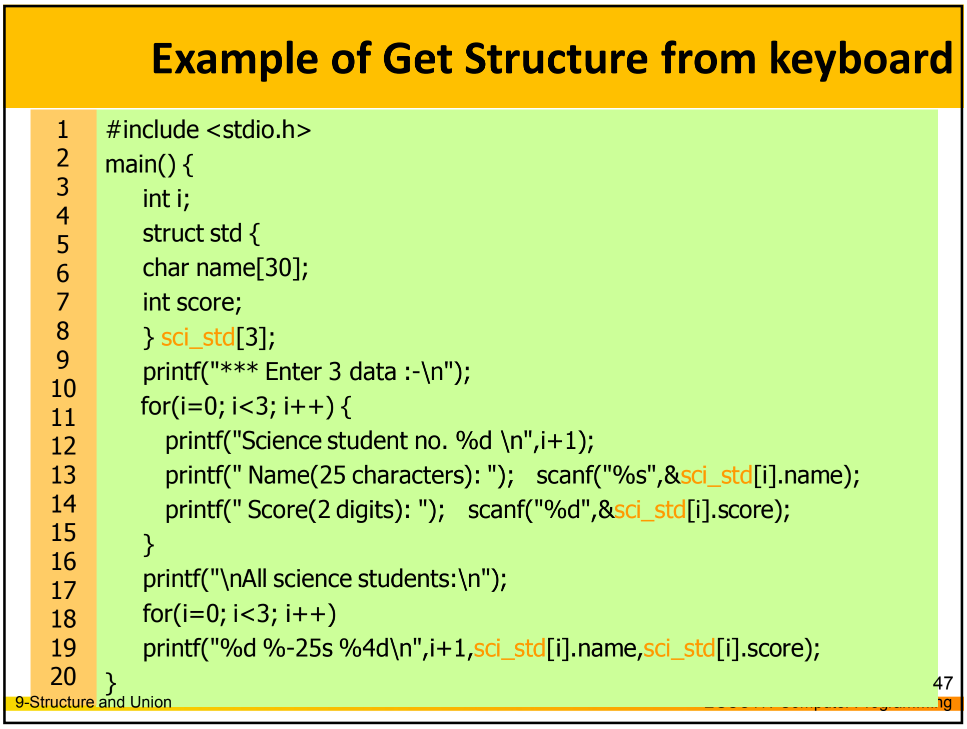


Example of Duplicate Value to Structure

```
#include <stdio.h>
#include <string.h>
main() {
     struct std {
     char name[30];
     int age;
     };   struct std sci_std = {"Pisut", 19};
          struct std math_std, eng_std;
          strcpy(math_std.name, sci_std.name);
          math_std.age = sci_std.age;
          eng_std = math_std;
printf("Science student: \n");
printf(" Name: %s Age: %d \n",sci_std.name,sci_std.age);
printf("Math student \n");
printf(" Name: %s Age: %d \n",math_std.name,math_std.age);
printf("English student :\n");
printf(" Name: %s Age: %d \n",eng_std.name,eng_std.age); }
```

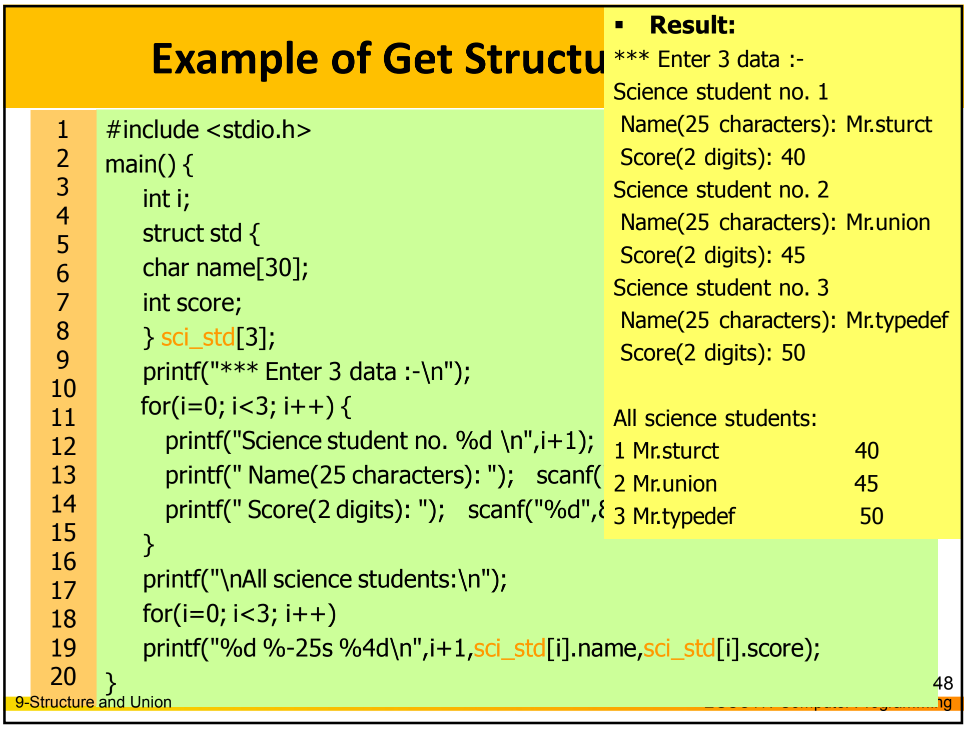
- **Result:**

```
Science student:
 Name: Pisut Age: 19
Math student
 Name: Pisut Age: 19
English student :
 Name: Pisut Age: 19
```



Example of Get Structure from keyboard

```
#include <stdio.h>
main() {
    int i;
    struct std {
    char name[30];
    int score;
    } sci_std[3];
    printf("*** Enter 3 data :-\n");
    for(i=0; i<3; i++) {
      printf("Science student no. %d \n",i+1);
      printf(" Name(25 characters): ");   scanf("%s",&sci_std[i].name);
      printf(" Score(2 digits): ");   scanf("%d",&sci_std[i].score);
    }
    printf("\nAll science students:\n");
    for(i=0; i<3; i++)
    printf("%d %-25s %4d\n",i+1,sci_std[i].name,sci_std[i].score);
}
```



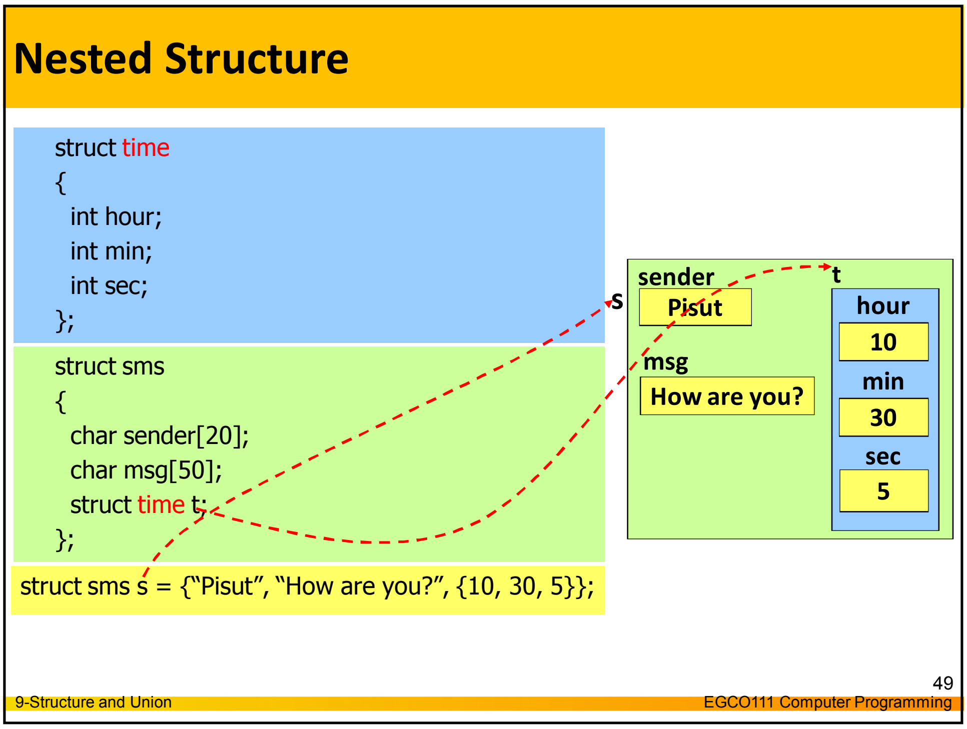
Example of Get Structu

```
#include <stdio.h>
main() {
    int i;
    struct std {
    char name[30];
    int score;
    } sci_std[3];
    printf("*** Enter 3 data :-\n");
    for(i=0; i<3; i++) {
      printf("Science student no. %d \n",i+1);
      printf(" Name(25 characters): ");   scanf(
      printf(" Score(2 digits): ");   scanf("%d",&
    }
    printf("\nAll science students:\n");
    for(i=0; i<3; i++)
    printf("%d %-25s %4d\n",i+1,sci_std[i].name,sci_std[i].score);
}
```

- **Result:**

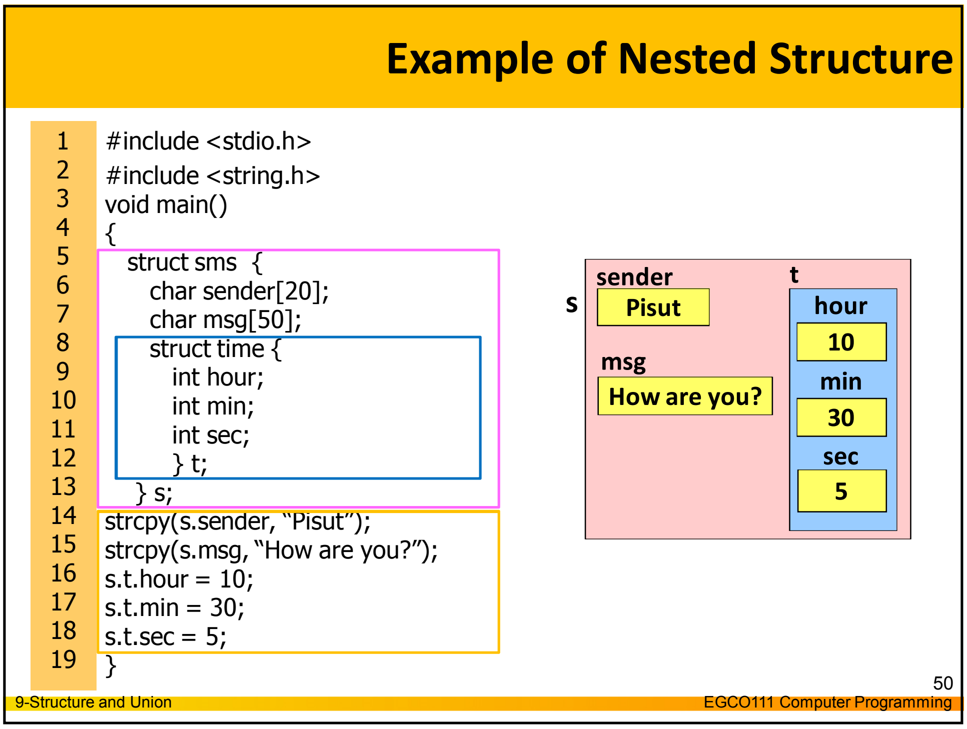
```
*** Enter 3 data :-
Science student no. 1
 Name(25 characters): Mr.sturct
 Score(2 digits): 40
Science student no. 2
 Name(25 characters): Mr.union
 Score(2 digits): 45
Science student no. 3
 Name(25 characters): Mr.typedef
 Score(2 digits): 50

All science students:
1 Mr.sturct                40
2 Mr.union                 45
3 Mr.typedef                50
```



Nested Structure

```
struct time
{
  int hour;
  int min;
  int sec;
};
struct sms
{
  char sender[20];
  char msg[50];
  struct time t;
};
struct sms s = {"Pisut", "How are you?", {10, 30, 5}};
```



Example of Nested Structure

```
#include <stdio.h>
#include <string.h>
void main()
{
   struct sms  {
      char sender[20];
      char msg[50];
      struct time {
         int hour;
         int min;
         int sec;
         } t;
    } s;
strcpy(s.sender, "Pisut");
strcpy(s.msg, "How are you?");
s.t.hour = 10;
s.t.min = 30;
s.t.sec = 5;
}
```

Typedef

```
struct student
   {
       char  name[30];
       float  score;
       float  grade;
   };
    struct student st[40];
```

```
typedef struct
   {
       char  name[30];
       float  score;
       float  grade;
   } student;
   student st[40];
```

- The keyword “stuct” is always used for declaration.
- This can be rewritten by using the keyword “typedef”.
- We will have a new data type. New variables can be declared without “struct” keyword.

Non-Typedef Vs. Typedef

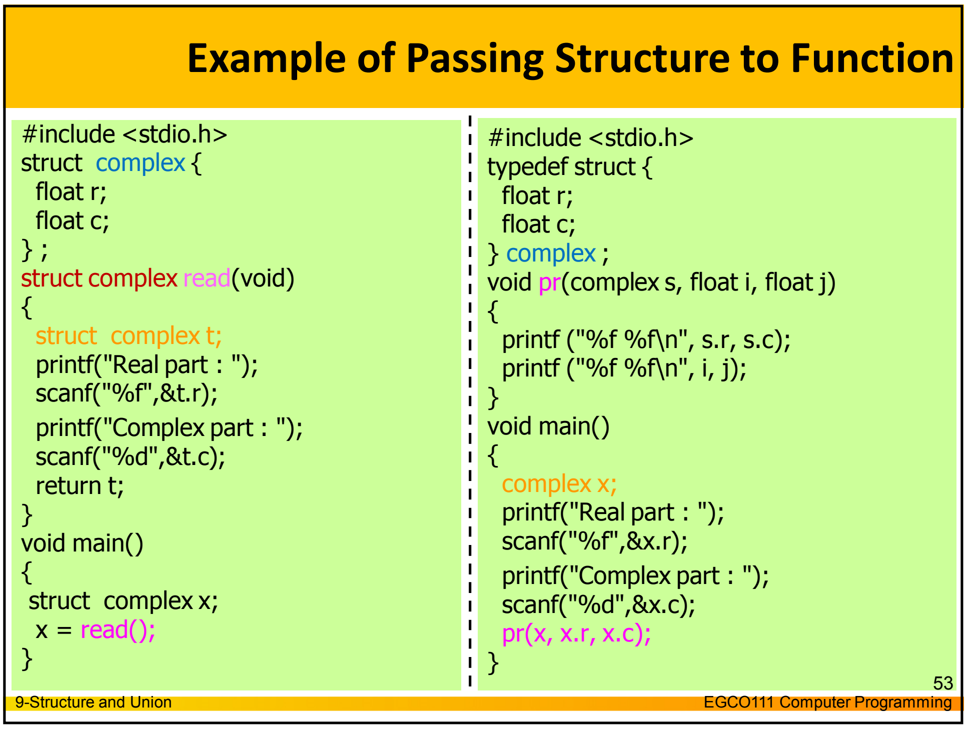
```
struct spoint {
       int a, b;
} p1;
```

```
struct {
       int a, b;
} p1;
```

```
struct spoint {
       int a, b;
};
struct spoint p1;
```

```
typedef struct {
     int a, b;
} tpoint;
tpoint p1;
```

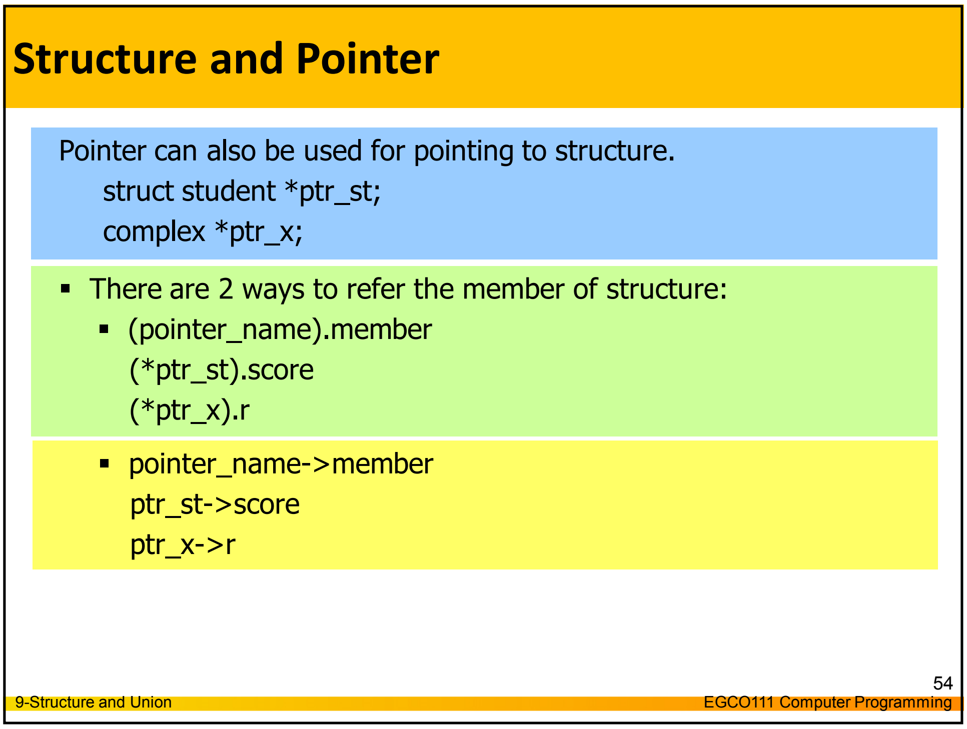
```
typedef struct spoint {
     int a, b;
} tpoint;
tpoint p1;
struct spoint p2;
```



Example of Passing Structure to Function

```
#include <stdio.h>
struct  complex {
  float r;
  float c;
} ;
struct complex read(void)
{
  struct  complex t;
  printf("Real part : ");
  scanf("%f",&t.r);
  printf("Complex part : ");
  scanf("%d",&t.c);
  return t;
}
void main()
{
 struct  complex x;
  x = read();
}
```

```
#include <stdio.h>
typedef struct {
  float r;
  float c;
} complex ;
void pr(complex s, float i, float j)
{
  printf ("%f %f\n", s.r, s.c);
  printf ("%f %f\n", i, j);
}
void main()
{
  complex x;
  printf("Real part : ");
  scanf("%f",&x.r);
  printf("Complex part : ");
  scanf("%d",&x.c);
  pr(x, x.r, x.c);
}
```



Structure and Pointer

Pointer can also be used for pointing to structure.

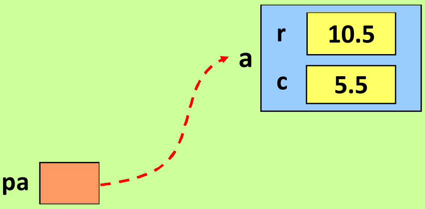
    struct student *ptr_st;
    complex *ptr_x;

- There are 2 ways to refer the member of structure:
  - (pointer_name).member
    (*ptr_st).score
    (*ptr_x).r
  - pointer_name->member
    ptr_st->score
    ptr_x->r

Example of Pointer to Structure

- **Example1**

```
#include <stdio.h>
main() {
typedef struct {
     double  r;
     double  c;
} complex;
 complex  a = {10.5,5.5};
 complex *pa;
 pa = &a;
 printf("%0.1f %0.1f\n",pa->r,pa->c);
 printf("%0.1f %0.1f\n",(*pa).r,(*pa).c);
}
```



a.r = (*pa).r = pa->r = 10.5
a.c = (*pa).c = pa->c = 5.5

- **Result:**

```
10.5 5.5
10.5 5.5
```

Question ?

- **Question1**

```
#include <stdio.h>
main() {
typedef struct {
     double  r;
     double  c;
} complex;
 complex  a = {10.5,5.5};
 complex *pa;
 printf("%0.1f %0.1f\n",(pa->r)+0.5,(pa->c)+5);
}
```

Question ?

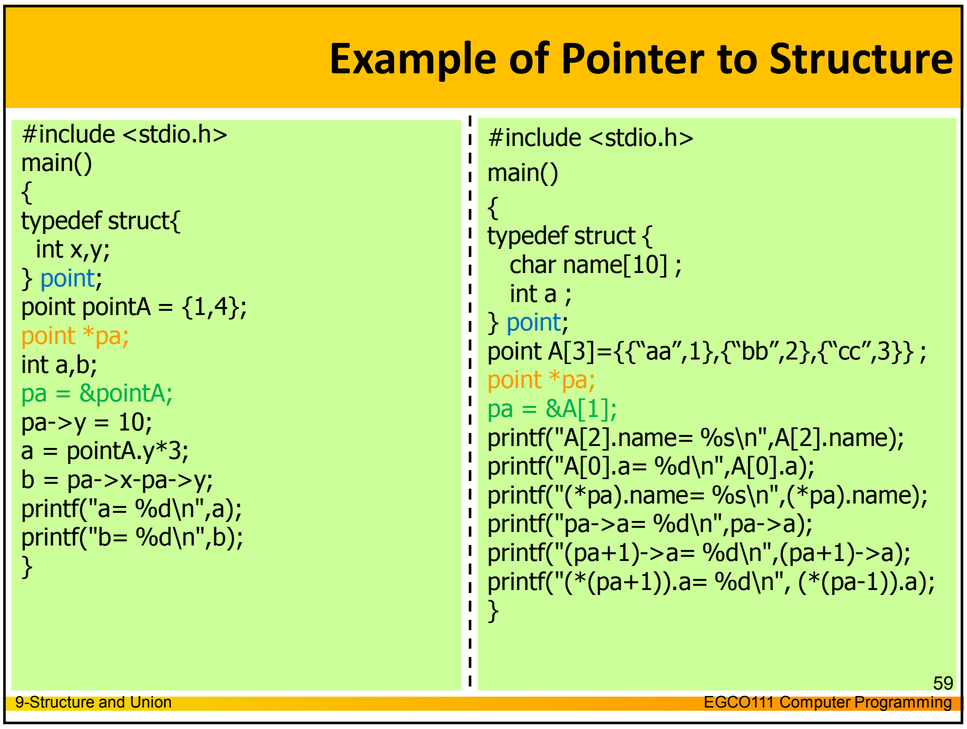
- **Question2**

```
#include <stdio.h>
main() {
struct complex {
     double  r;
     double  c;
} *pa;
 struct complex  a = {10.5,5.5};
 complex;
 pa = &a;
 printf("%0.1f %0.1f\n",(*pa).r,(*pa).c);
}
```

Question ?

- **Question3**

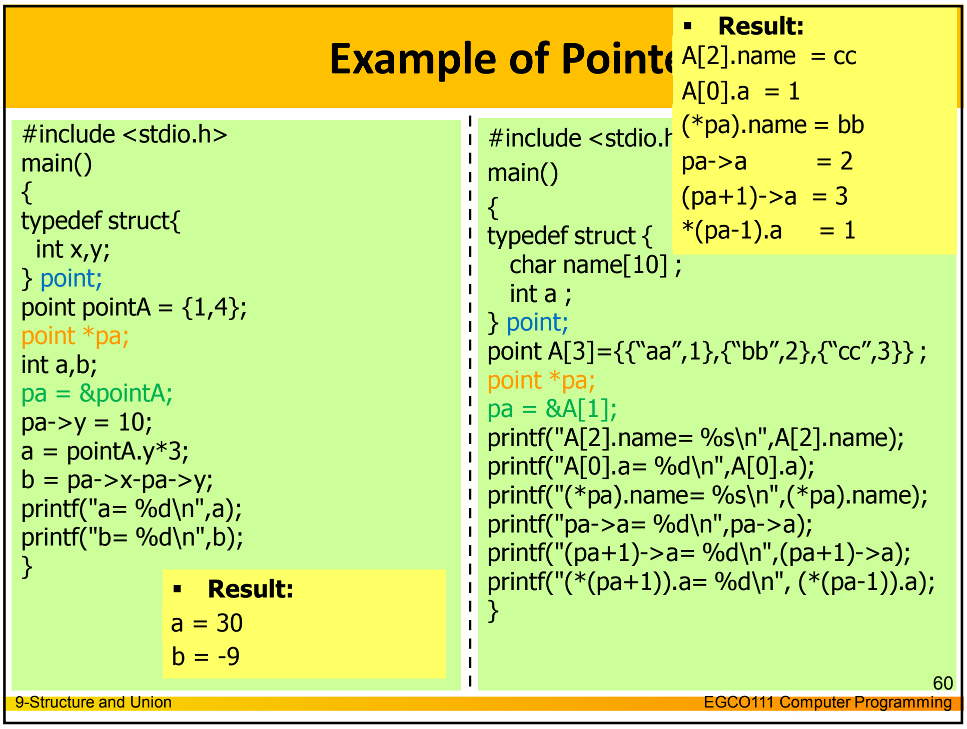
```
#include <stdio.h>
struct complex {
     double  r;
     double  c;
} *pa;
main() {
struct complex  a = {10.5,5.5};
 complex;
 printf("%0.1f %0.1f\n",(pa->r)+0.5,(pa->c)+5);
}
```



Example of Pointer to Structure

```
#include <stdio.h>
main()
{
typedef struct{
  int x,y;
} point;
point pointA = {1,4};
point *pa;
int a,b;
pa = &pointA;
pa->y = 10;
a = pointA.y*3;
b = pa->x-pa->y;
printf("a= %d\n",a);
printf("b= %d\n",b);
}
```

```
#include <stdio.h>
main()
{
typedef struct {
   char name[10] ;
   int a ;
} point;
point A[3]={{“aa”,1},{“bb”,2},{“cc”,3}} ;
point *pa;
pa = &A[1];
printf("A[2].name= %s\n",A[2].name);
printf("A[0].a= %d\n",A[0].a);
printf("(*pa).name= %s\n",(*pa).name);
printf("pa->a= %d\n",pa->a);
printf("(pa+1)->a= %d\n",(pa+1)->a);
printf("(*(pa+1)).a= %d\n", (*(pa-1)).a);
}
```



Example of Pointe

- **Result:**

```
A[2].name  = cc
A[0].a  = 1
(*pa).name = bb
pa->a         = 2
(pa+1)->a  = 3
*(pa-1).a     = 1
```

```
#include <stdio.h>
main()
{
typedef struct{
  int x,y;
} point;
point pointA = {1,4};
point *pa;
int a,b;
pa = &pointA;
pa->y = 10;
a = pointA.y*3;
b = pa->x-pa->y;
printf("a= %d\n",a);
printf("b= %d\n",b);
}
```

- **Result:**

```
a = 30
b = -9
```

```
#include <stdio.h
main()
{
typedef struct {
   char name[10] ;
   int a ;
} point;
point A[3]={{“aa”,1},{“bb”,2},{“cc”,3}} ;
point *pa;
pa = &A[1];
printf("A[2].name= %s\n",A[2].name);
printf("A[0].a= %d\n",A[0].a);
printf("(*pa).name= %s\n",(*pa).name);
printf("pa->a= %d\n",pa->a);
printf("(pa+1)->a= %d\n",(pa+1)->a);
printf("(*(pa+1)).a= %d\n", (*(pa-1)).a);
}
```

Function of Pointer VS. Function of Array

```
#include <stdio.h>
typedef struct {
  int r,c;
} complex;
void read(complex *px)
{
  printf("Real part : ");
  scanf("%d",&(*px).r);
  printf("Complex part : ");
  scanf("%d",&px->c);
}
void main()
{
  complex x;
  read(&x);
  printf("%d %d",x.r,x.c);
}
```

```
#include <stdio.h>
typedef struct {
  int r,c;
} complex;
void read(complex *px)
{
  printf("Real part : ");
  scanf("%d",&(*px).r);
  printf("Complex part : ");
  scanf("%d",&px->c);
}
void main()
{
  complex x[2];
  read(x);
  printf("%d %d",x[0].r,x[0].c);
  printf("%d %d",x[1].r,x[1].c);
}
```

Thanks for your attention