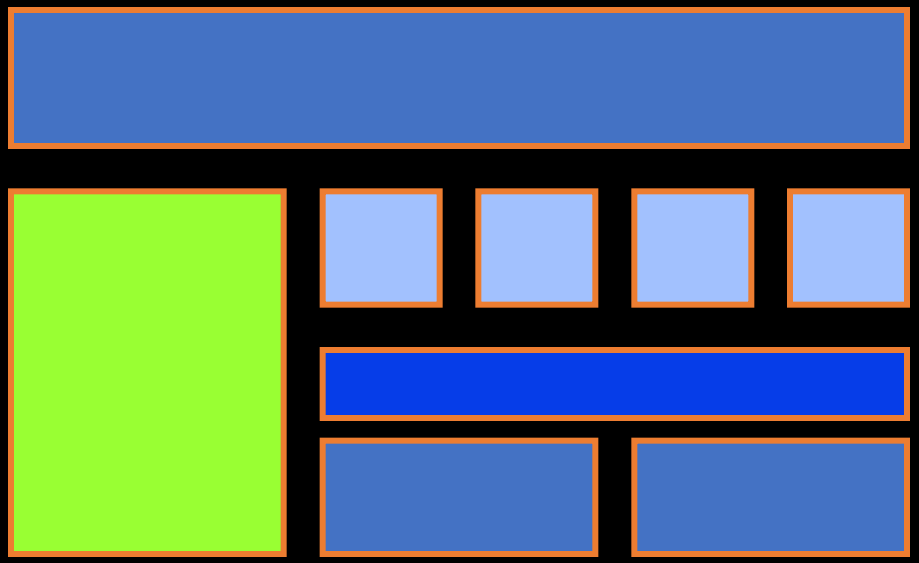
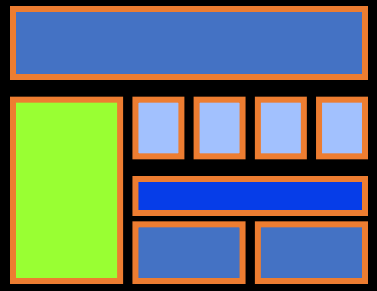
Programming



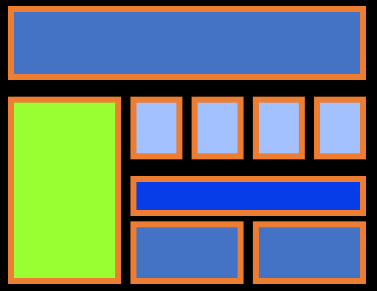
Structures

Structures



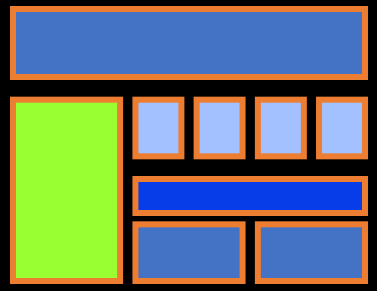
- A Structure is a collection of related data items, possibly of different types.
- A structure type in C++ is called struct.
- A struct is heterogeneous in that it can be composed of data of different types.
- In contrast, array is homogeneous since it can contain only data of the same type.

Structures



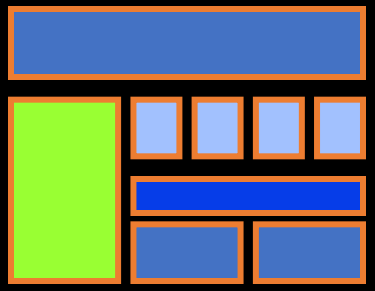
- Structures hold data that belong **together**.
- Examples:
  - Student record: student id, name, major, gender, start year, …
  - Bank account: account number, name, currency, balance, …
  - Address book: name, address, telephone number, …
- In database applications, structures are called records.

Structures



- Individual components of a struct type are called members (or fields).
- Members can be of different types (simple, array or struct).
- A struct is named as a whole while individual members are named using field identifiers.
- Complex data structures can be formed by defining arrays of structs.

struct basics



- Definition of a structure:

```
struct <struct-type>{
        <type> <identifier_list>;
        <type> <identifier_list>;
        ...
} ;
```

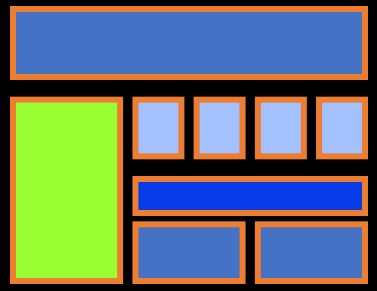
Each identifier defines a member of the structure.

- Example:

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

The "Date" structure has 3 members, day, month & year.

struct examples



- Example:

```
struct StudentInfo{
        int Id;
        int age;
        char Gender;
        double CGA;
};
```

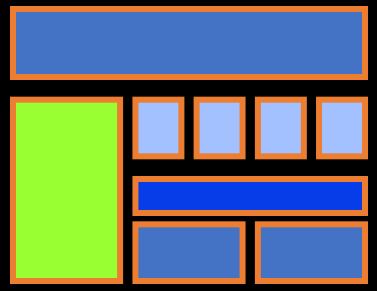
The “StudentInfo” structure has 4 members of different types.

- Example:

```
struct StudentGrade{
        char Name[15];
        char Course[9];
        int Lab[5];
        int Homework[3];
        int Exam[2];
};
```

The “StudentGrade” structure has 5 members of different array types.

struct examples



- Example:

```
struct BankAccount{
        char Name[15];
        int AcountNo[10];
        double balance;
        Date Birthday;
};
```

The “BankAcount” structure has simple, array and structure types as members.

- Example:

```
struct StudentRecord{
        char Name[15];
        int Id;
        char Dept[5];
        char Gender;
};
```

The “StudentRecord” structure has 4 members.

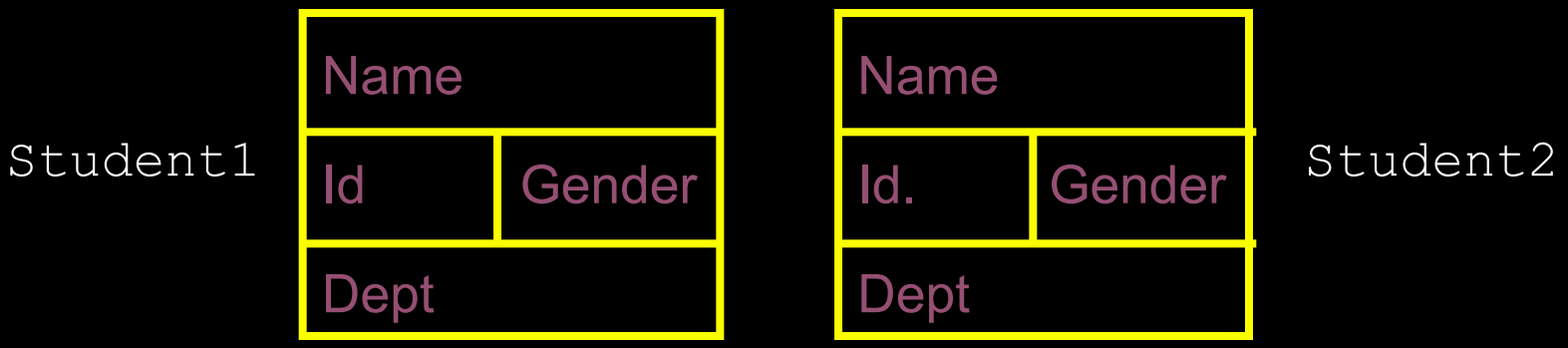
struct basics

- Declaration of a variable of struct type:

```
<struct-type> <identifier_list>;
```

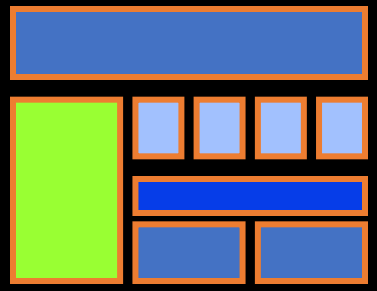
- Example:

```
StudentRecord Student1, Student2;
```



Student1 and **Student2** are variables of **StudentRecord** type.

Ex. 1: struct basics

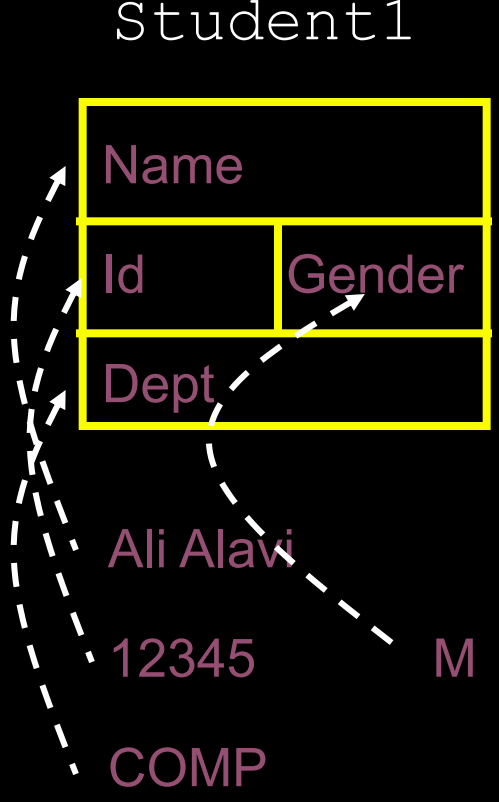


- The members of a struct type variable are accessed with the dot (.) operator:

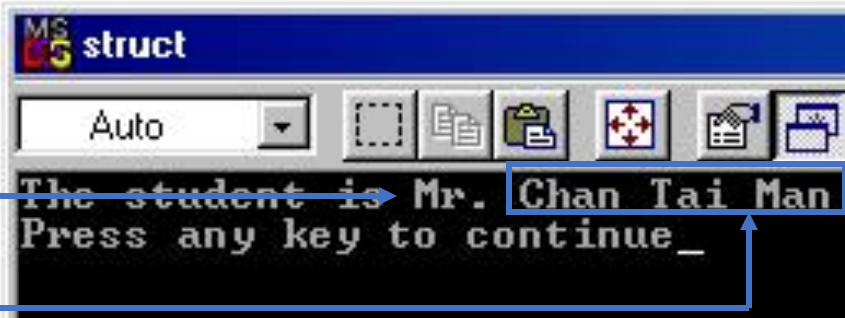
```
<struct-variable>.<member_name>;
```

- Example:

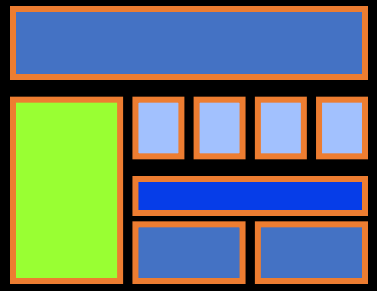
```
strcpy(Student1.Name, "Ali Alavi");
Student1.Id = 12345;
strcpy(Student1.Dept, "COMP");
Student1.gender = 'M';
cout << "The student is ";
switch (Student1.gender){
     case 'F': cout << "Ms. "; break;
     case 'M': cout << "Mr. "; break;
}
cout << Student1.Name << endl;
```



```
#include <string.h>

struct StudentRecord {
        char Name[22];
        int Id;
        char Dept[22];
        char gender;
};

int main()  {
    StudentRecord Student1;

    strcpy(Student1.Name, "Chan Tai Man");
    Student1.Id = 12345;
    strcpy(Student1.Dept, "COMP");
    Student1.gender = 'M';

    cout << "The student is ";
    switch (Student1.gender){
        case 'F': cout << "Ms. "; break;
        case 'M': cout << "Mr. "; break;
    }
    cout << Student1.Name << endl;
    return 0;
}
```

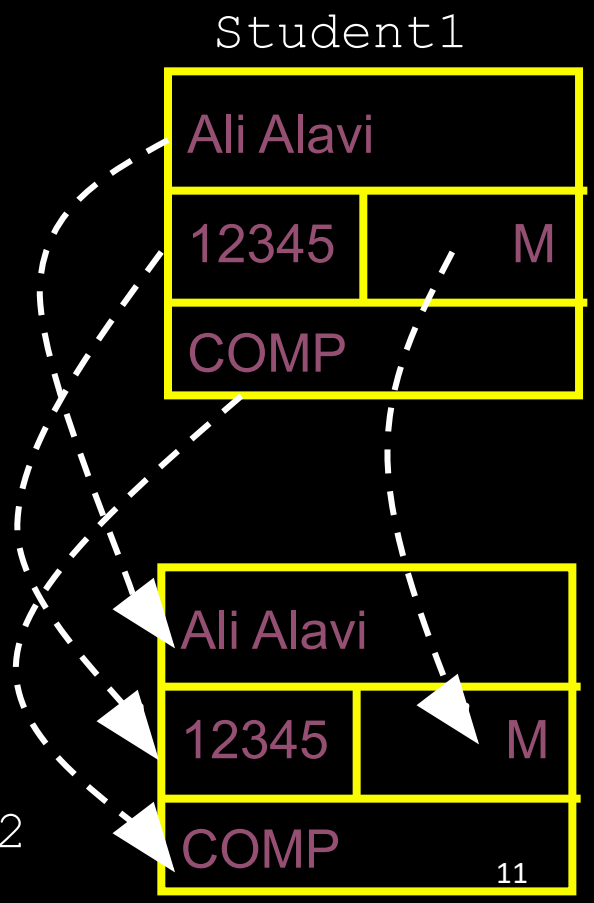


Ex. 2: struct-to-struct assignment

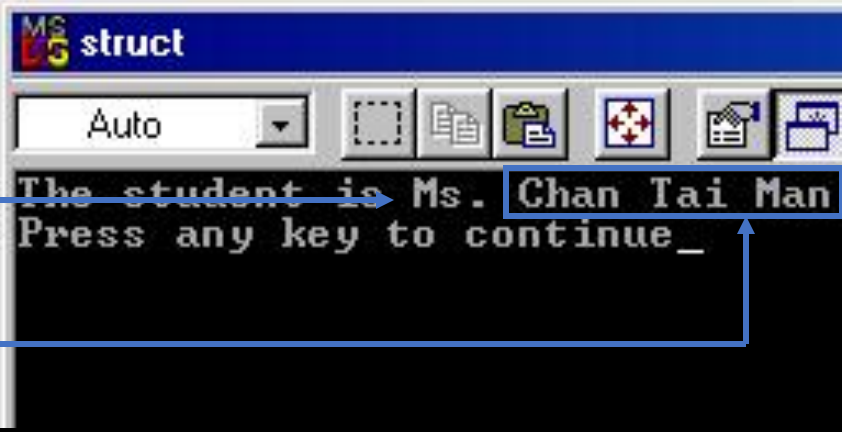


- The values contained in one struct type variable can be assigned to another variable of the same struct type.
- Example:

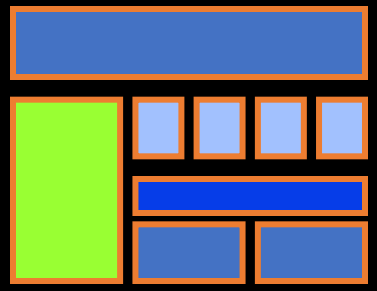
```
strcpy(Student1.Name,
        "Ali Alavi");
Student1.Id = 12345;
strcpy(Student1.Dept, "COMP");
Student1.gender = 'M';

Student2 = Student1;
```



```
struct StudentRecord {
        char Name[22];
        int Id;
        char Dept[22];
        char gender;
};

int main()  {
    StudentRecord Student1, Student2;

    strcpy(Student1.Name, "Chan Tai Man");
    Student1.Id = 12345;
    strcpy(Student1.Dept, "COMP");
    Student1.gender = 'M';

    Student2 = Student1;
    Student2.gender = 'F';

    cout << "The student is ";
    switch (Student2.gender){
        case 'F': cout << "Ms. "; break;
        case 'M': cout << "Mr. "; break;
    }
    cout << Student2.Name << endl;
    return 0;
}
```

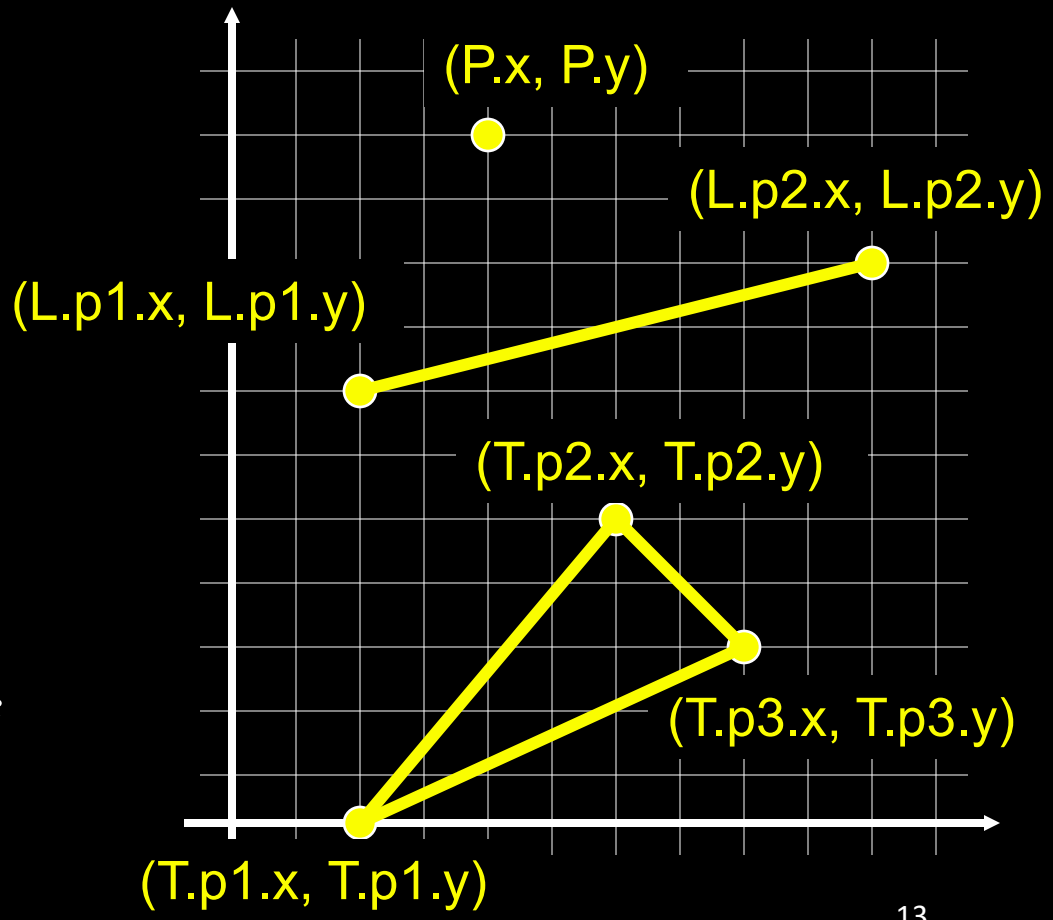


Ex. 3-5: Nested structures

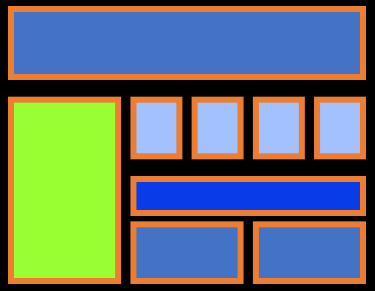


- We can nest structures inside structures.
- Examples:

```
struct point{
     double x, y;
};
point P;

struct line{
     point p1, p2;
};
line L;

struct triangle{
     point p1, p2, p3;
};
triangle T;
```

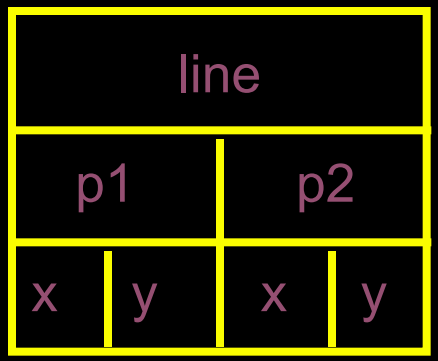


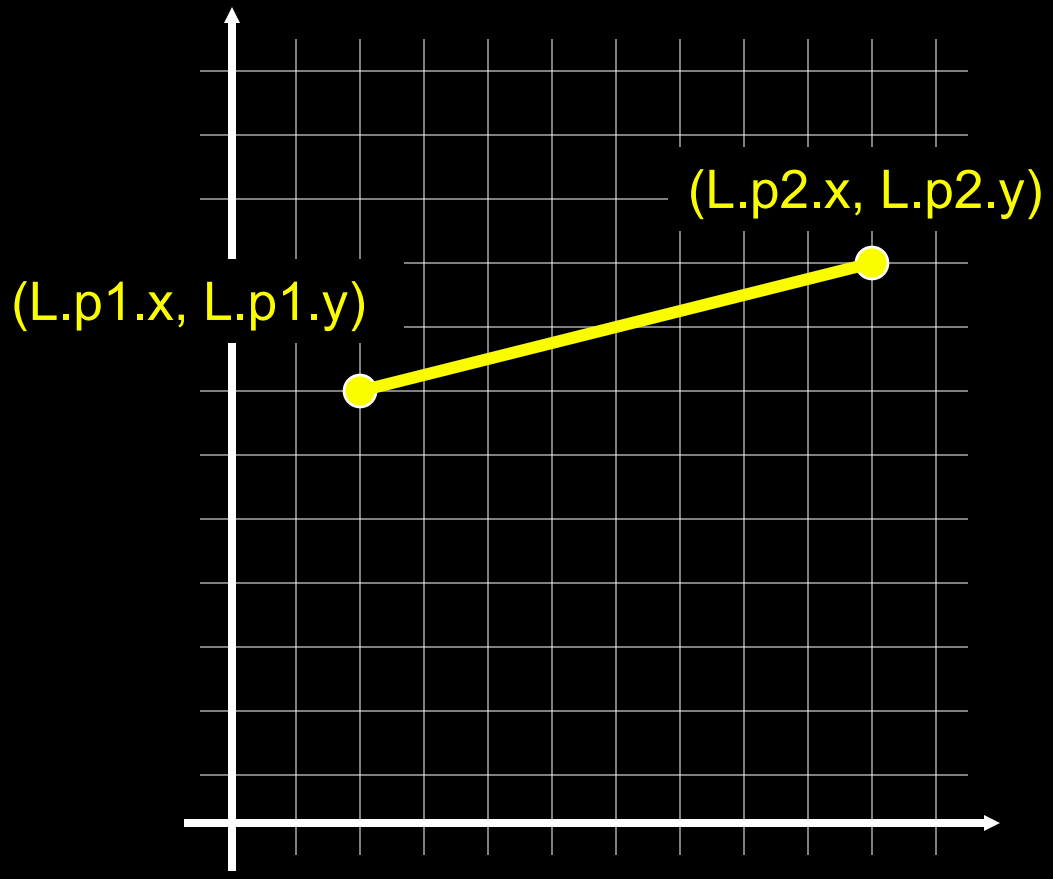
Ex. 3-5: Nested structures



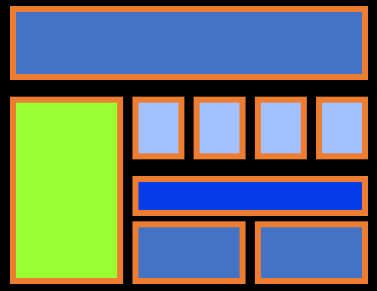
- We can nest structures inside structures.

```
struct line{
     point p1, p2;
};
line L;
```



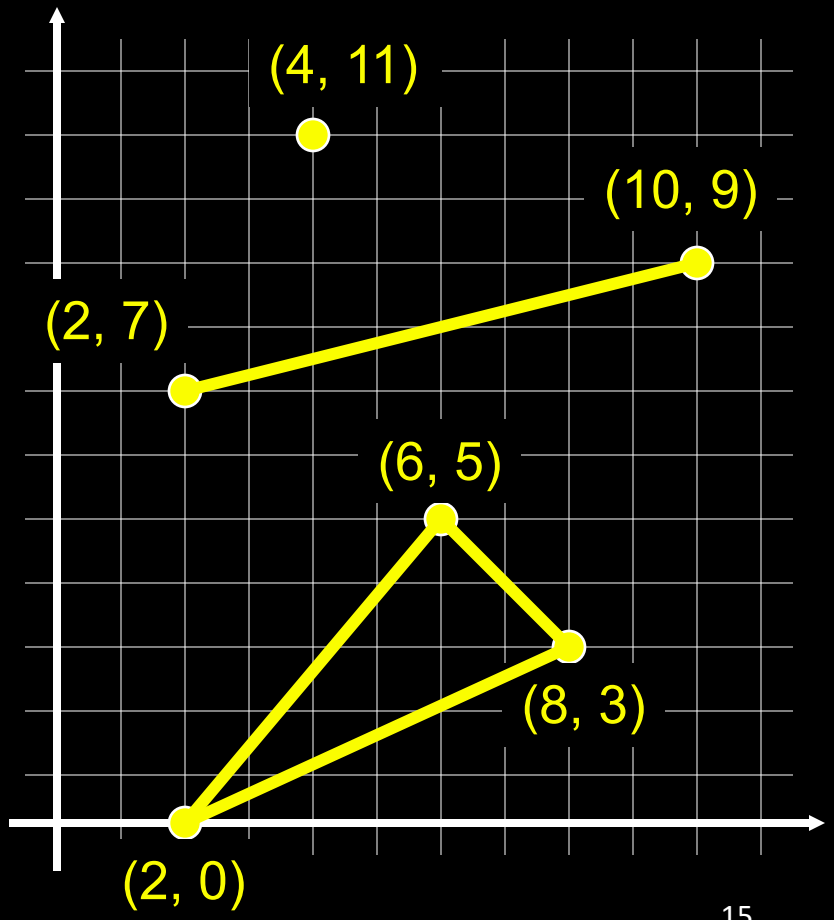


Ex. 3-5: Nested structures



- Assign values to the variables **`P`**, **`L`**, and **`T`** using the picture:

```
point P;
line L;
triangle T;
```

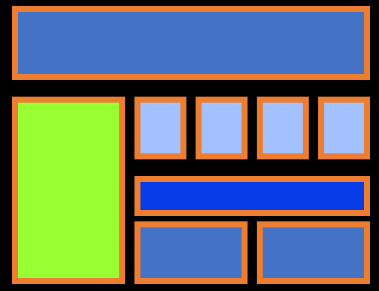


Ex. 3: Graph a point

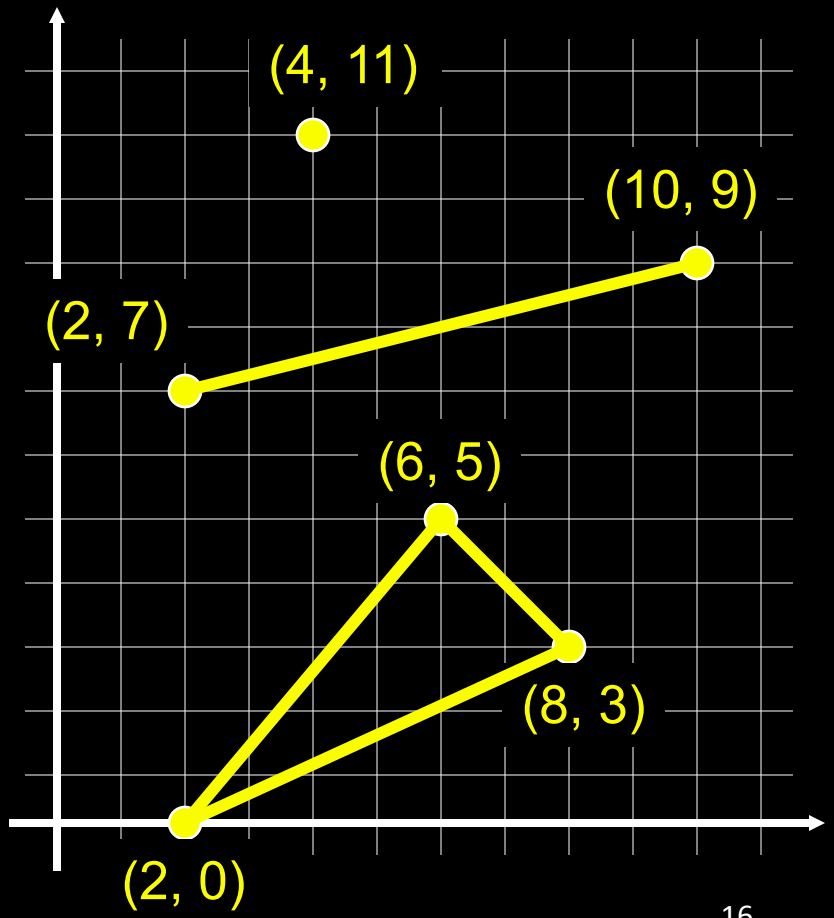
Ex. 4: Graph a line

Ex. 5: Graph a triangle

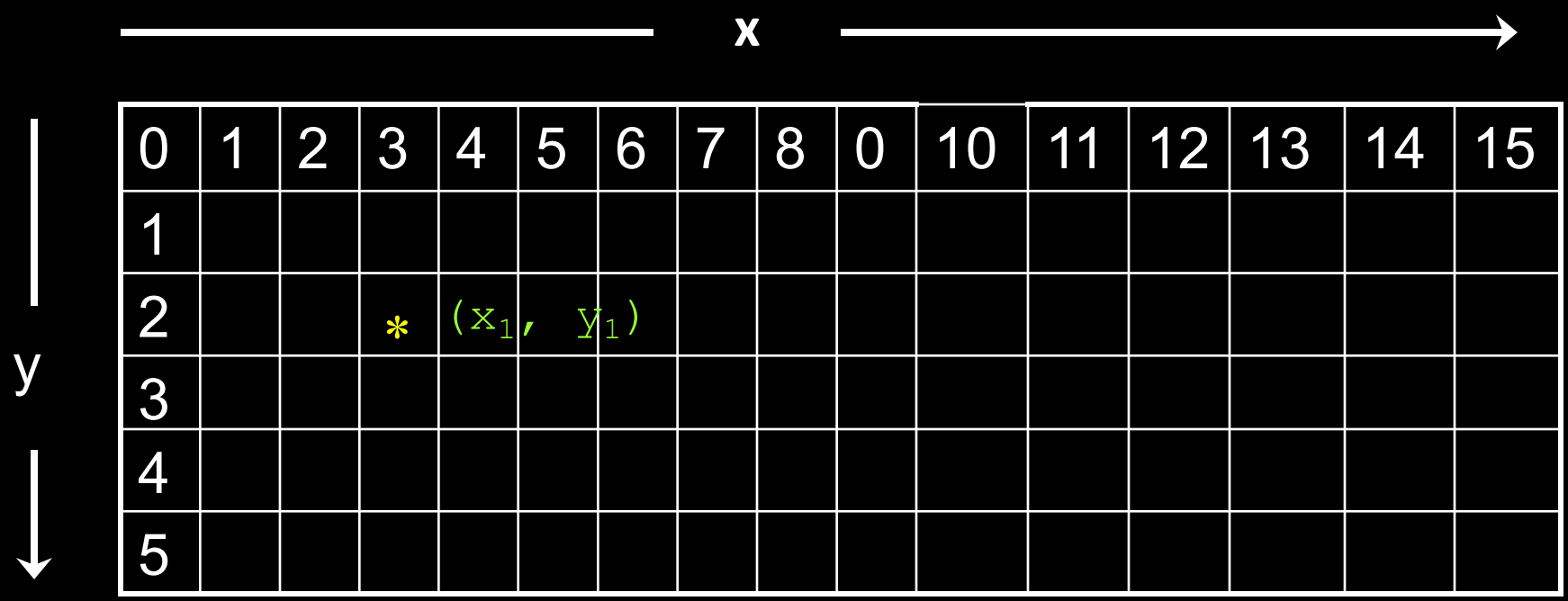
Ex. 3-5: Nested structures



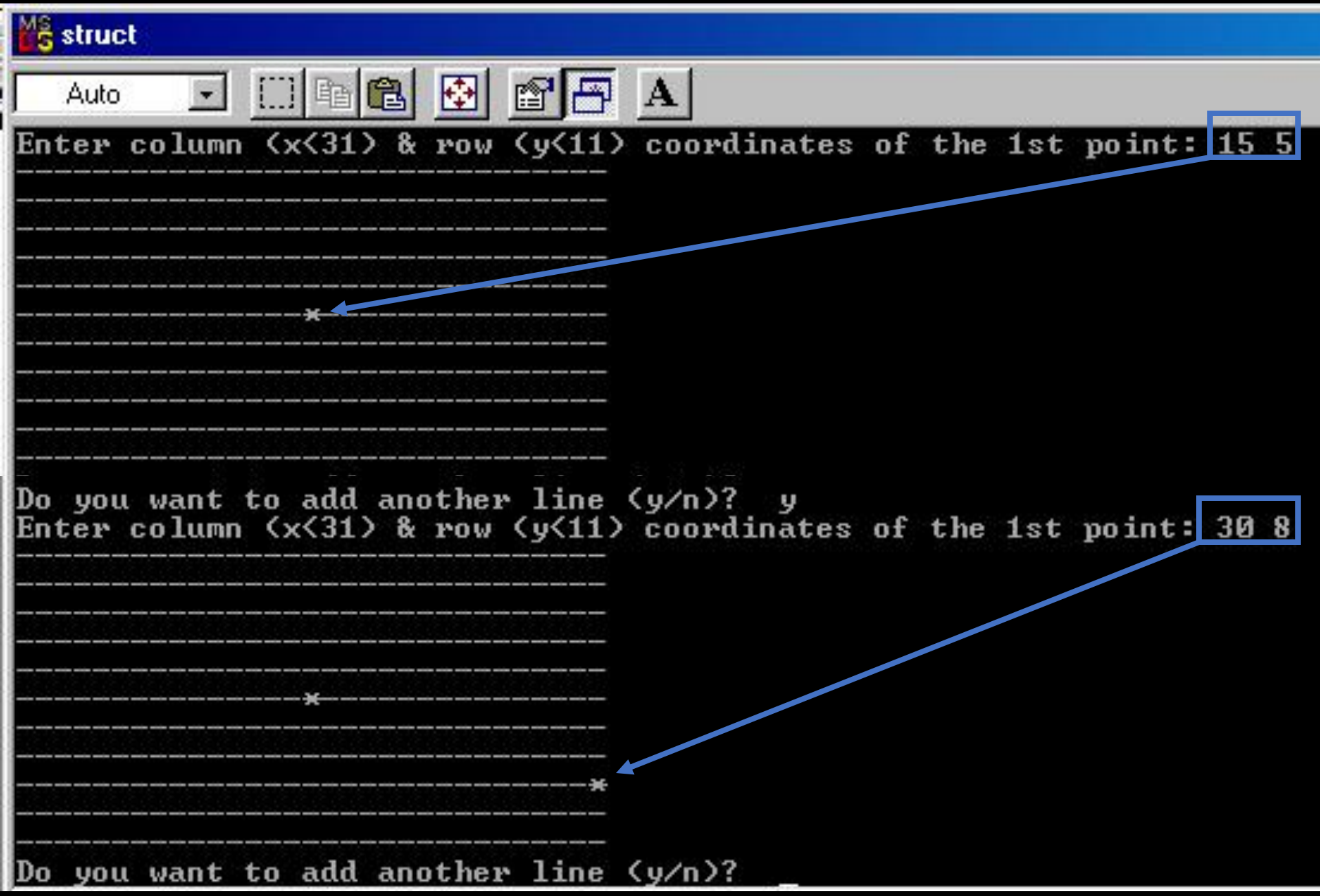
```
point P;
line L;
triangle T;

P.x = 4;

P.y = 11;

 L.p1.x = 2;
 L.p1.y = 7;
 L.p2.x = 10;
 L.p2.y = 9;

 T.p1.x = 2;
 T.p1.y = 0;
 T.p2.x = 6;
 T.p2.y = 5;
 T.p3.x = 8;
 T.p3.y = 3;
```



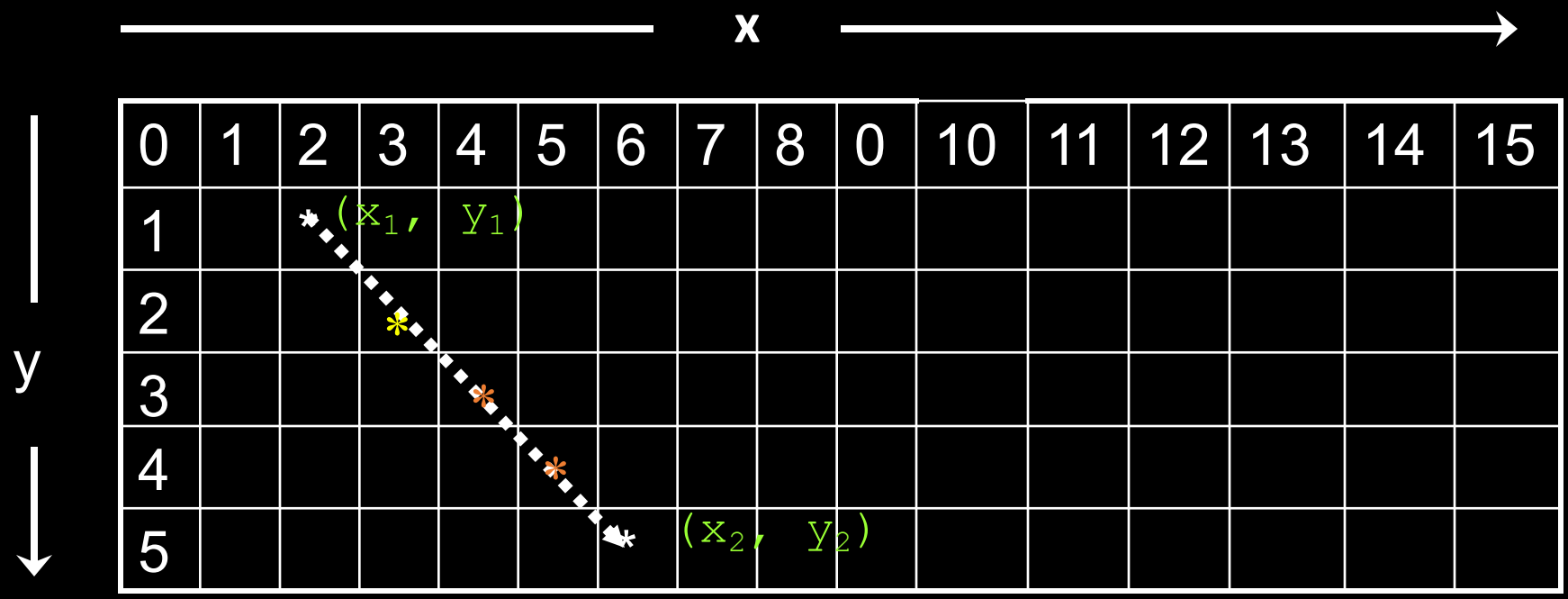
Ex. 3: Graphing a Point



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

void user_input(point&);
void graph_point(char grid[NUMBER_ROWS][NUMBER_COLS], point);
void print_grid(char grid[NUMBER_ROWS][NUMBER_COLS]);
void set_background(char grid[][NUMBER_COLS]);

void user_input(point& P){ // pass by reference
  // get user input and check that it is on the grid
        do{
    cout << "Enter column (x<" << NUMBER_COLS << ") & row (y<"
          << NUMBER_ROWS <<") of the 1st point: ";
     cin >> P.x >> P.y;
   } while ((P.y<0) || (P.y >= NUMBER_ROWS) ||
              (P.x<0) || (P.x >= NUMBER_COLS));
}
// Put a point on the grid
void graph_point(char grid[][NUMBER_COLS], point P){
   grid[P.y][P.x] = '*';
}
```


struct
Auto
Enter column (x<31) & row (y<11) coordinates of the 1st point: 15 5
Do you want to add another line (y/n)? y
Enter column (x<31) & row (y<11) coordinates of the 1st point: 30 8
Do you want to add another line (y/n)?

Ex. 4: Trajectory Between 2 Points



- The equation of a line going through two points (x_1, y_1) & (x_2, y_2) can be represented by:

$$(y-y_1)/(x-x_1) = (y_2-y_1)/(x_2-x_1)$$

$$\text{or } y = ((y_2-y1)/(x_2-x_1))\ (x-x_1) + y_1$$

where $(y_2-y_1)/(x_2-x_1)$ is called the slope.

```
// use struct to graph line between two points
#include <ctype>                        // for tolower
#include <iostream>
using namespace std;

int const NUMBER_ROWS = 11;
int const NUMBER_COLS = 31;

struct point {int x, y;};
struct line {point p1, p2;};

void user_input (line&);
void graph_line (char grid[NUMBER_ROWS][NUMBER_COLS], line);
void print_grid(char grid[NUMBER_ROWS][NUMBER_COLS]);
void set_background (char grid[][NUMBER_COLS]);
```

```
// Graph line between two points
int main(){
  // set an array for the grid
        char grid[NUMBER_ROWS][NUMBER_COLS];
  line line1;
  int row1=0, col1=0, row2=0, col2=0;
        char do_another;

        // function call to fill background of grid with '-'
        set_background(grid);
  do{             // do-while loop allows multiple lines
        // get user input
        user_input(line1);
        // put '*' into array to graph the line(s) on the grid
        graph_line(grid, line1);
        // print the grid from the array to the screen
        print_grid(grid);
        cout << "Do you want to add another line (y/n)?  ";
        cin >> do_another;
        do_another = tolower(do_another);
  } while (do_another == 'y');
  return 0;
}
```

Ex. 4: More on Graphing Line

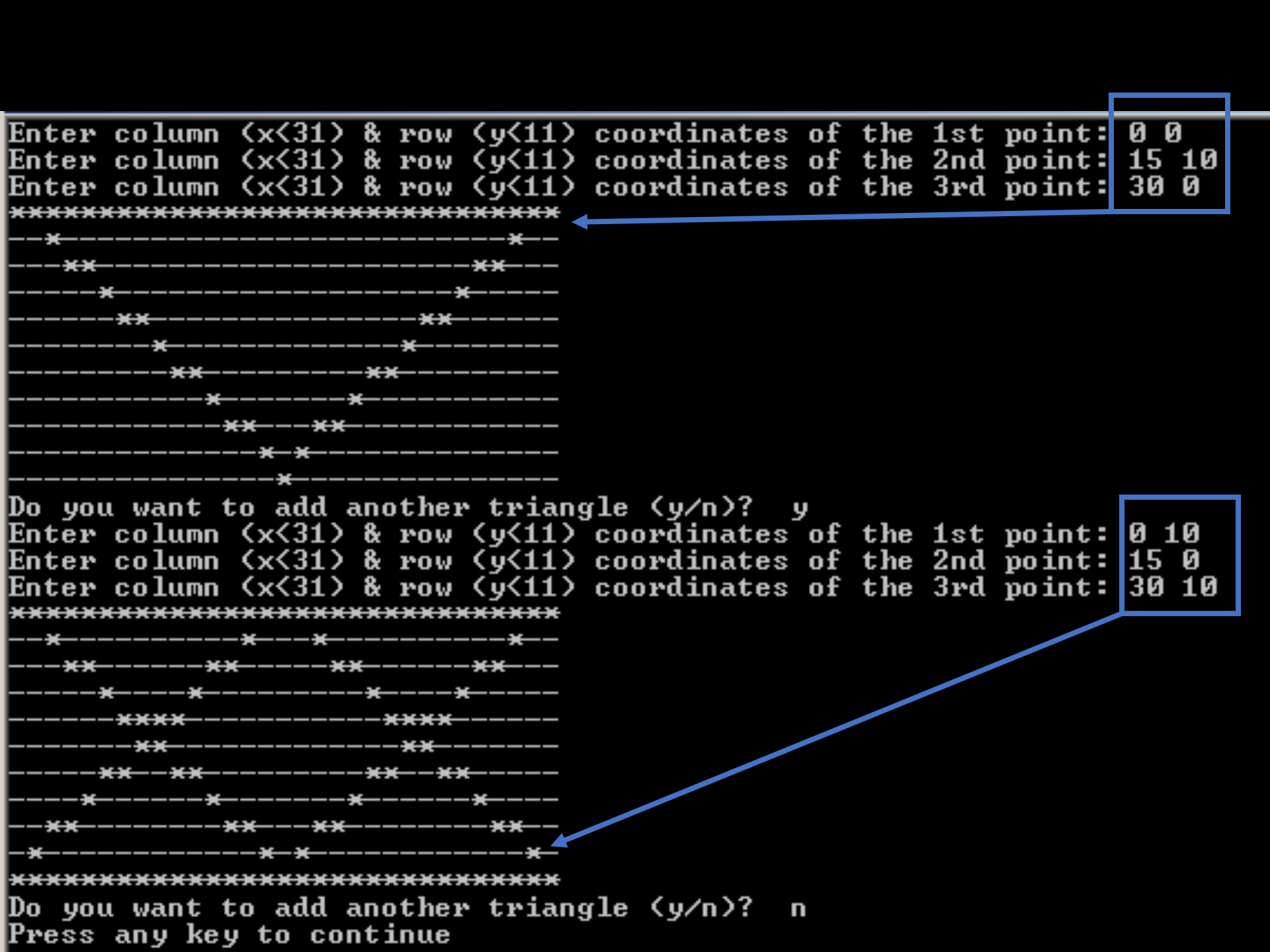
```
//A function to get user input and check that it is on the grid.
void user_input(line& line1){
    do{
          cout << "Enter column (x<" << NUMBER_COLS
               << ") & row (y<" << NUMBER_ROWS
               << ") coordinates of the 1st point: ";
          cin >> line1.p1.x >> line1.p1.y;
     } while ((line1.p1.y<0) ||
              (line1.p1.y>=NUMBER_ROWS) ||
              (line1.p1.x<0) ||
              (line1.p1.x >= NUMBER_COLS));
      // use another do-while loop for the 2nd point, col2 and row2
}
```

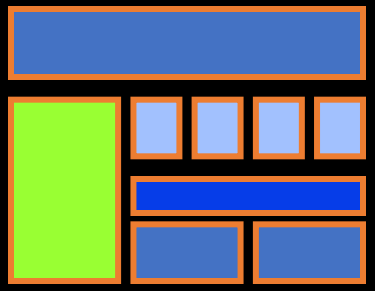
Ex. 4: More on Graphing Line

```
void graph_line(char grid[][NUMBER_COLS], line line1){
   int row, col;
   double rise, run, slope;
   // one point
   if((line1.p1.y==line1.p2.y)&&(line1.p1.x==line2.p2.x))
        grid[line1.p1.y][line1.p1.x] = '*';
   else if(line1.p2.x==line1.p1.x){  // infinite slope
        if (line1.p1.y < line1.p2.y){
            for(row=line1.p1.y; row <= line1.p2.y; row++)
                grid[row][line1.p1.x] = '*';
        }
        else{
            for(row=line1.p1.y; row >= line1.p2.y; row--)
                grid[row][line1.p1.x] = '*';
        }
   }
```

Ex. 4: More on Graphing Line

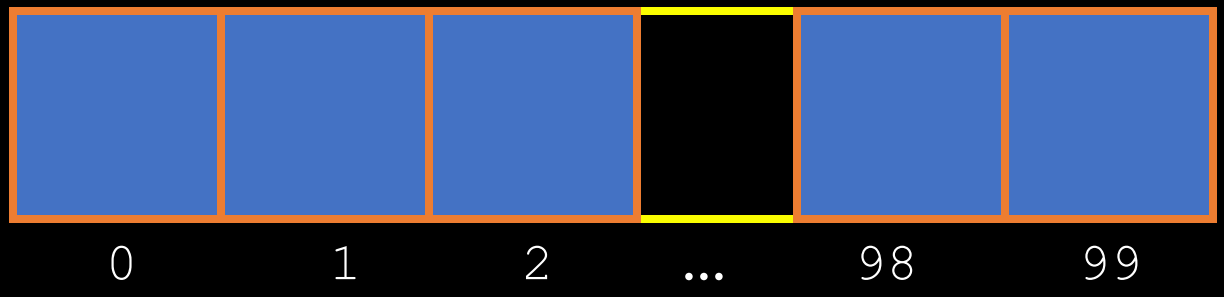
```
    else{
       rise=line1.p2.y-line1.p1.y;  run=line1.p2.x-line1.p1.x;
       slope = (double)rise / run;   // run cannot = 0
       if (run >0){
             for(col = line1.p1.x; col <= line1.p2.x; col++){
                 // line1.p1.y is offset for starting point
                 row=(int)(slope*(col-line1.p1.x)+line1.p1.y);
                 grid[row][col] = '*';
             }
       }
       else{
             for(col=line1.p1.x; col >= line1.p2.x; col--){
                 row=(int)(slope*(col-line1.p1.x)+line1.p1.y);
                 grid[row][col] = '*';
             }
       }
    }
}
```


Enter column (x<31) & row (y<11) coordinates of the 1st point: 0 0
Enter column (x<31) & row (y<11) coordinates of the 2nd point: 15 10
Enter column (x<31) & row (y<11) coordinates of the 3rd point: 30 0
Do you want to add another triangle (y/n)? y
Enter column (x<31) & row (y<11) coordinates of the 1st point: 0 10
Enter column (x<31) & row (y<11) coordinates of the 2nd point: 15 0
Enter column (x<31) & row (y<11) coordinates of the 3rd point: 30 10
Do you want to add another triangle (y/n)? n
Press any key to continue

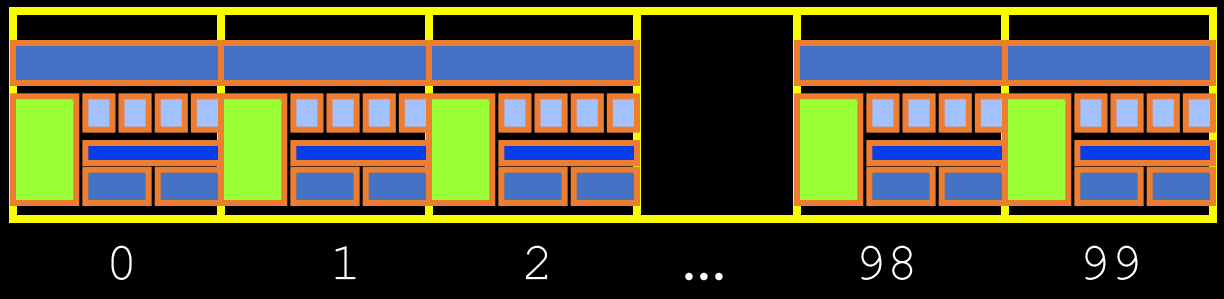


Arrays of structures

- An ordinary array: One type of data



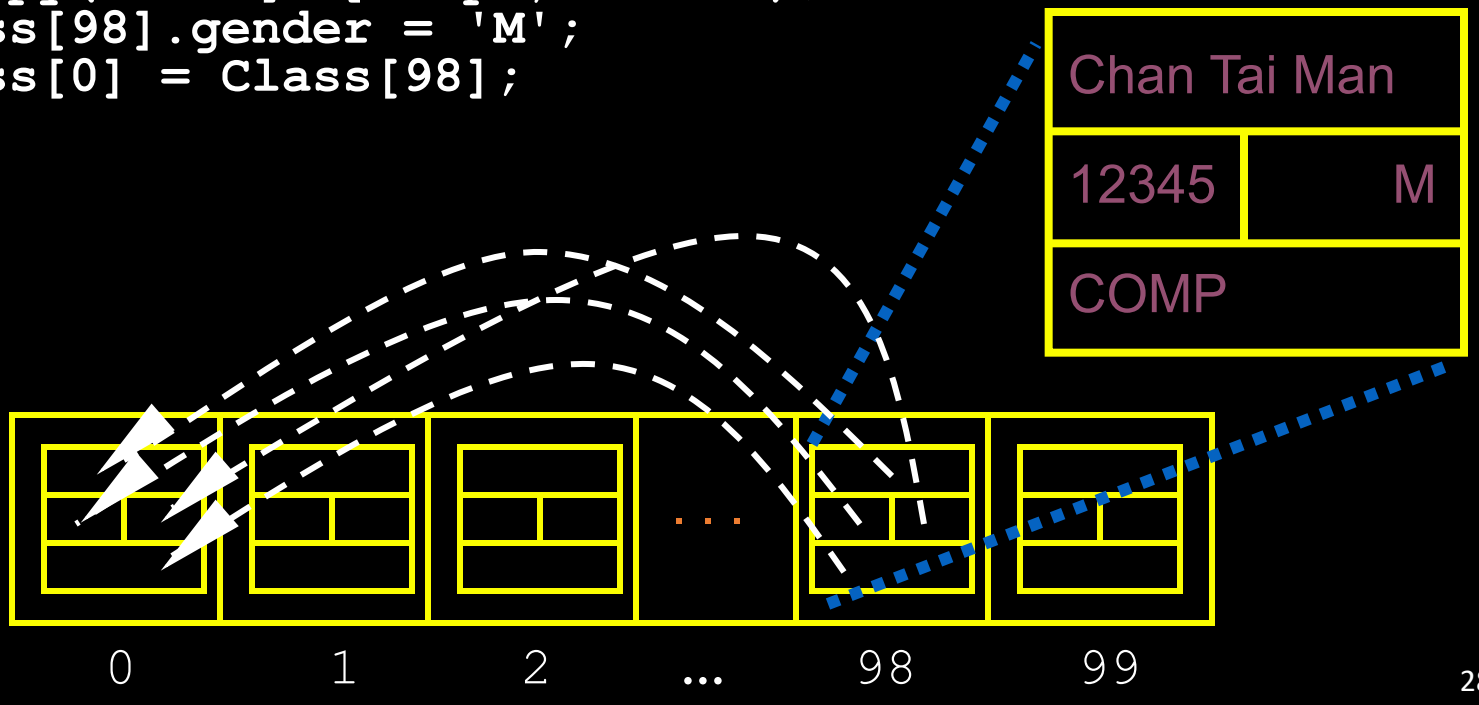
- An array of structs: Multiple types of data in each array element.



Arrays of structures

- We often use arrays of structures.
- Example:

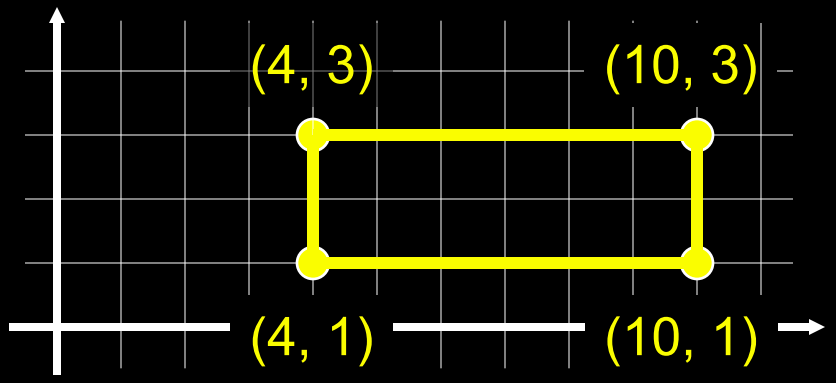
```
StudentRecord Class[100];
strcpy(Class[98].Name, "Chan Tai Man");
Class[98].Id = 12345;
strcpy(Class[98].Dept, "COMP");
Class[98].gender = 'M';
Class[0] = Class[98];
```



Arrays inside structures

- We can use arrays inside structures.
- Example:

```
struct square{
    point vertex[4];
};
square Sq;
```



- Assign values to **Sq** using the given square