Object Oriented Programming

Shaobai Kan

chapter 22

Outline

Outline

- Array of struct data types

Outline

- Array of struct data types
- Passing struct data to functions

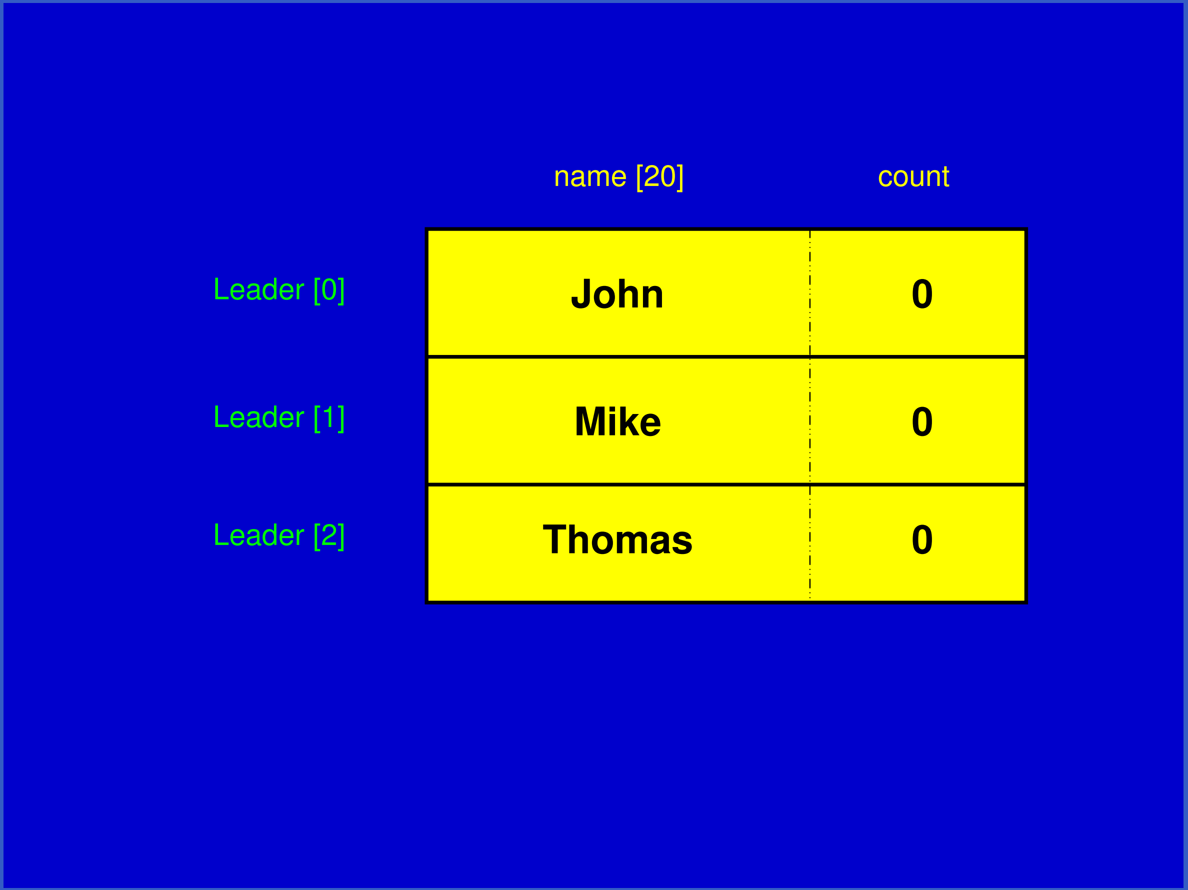
Array of struct data types

Example 1: array of struct data types

Array of `struct` data types I

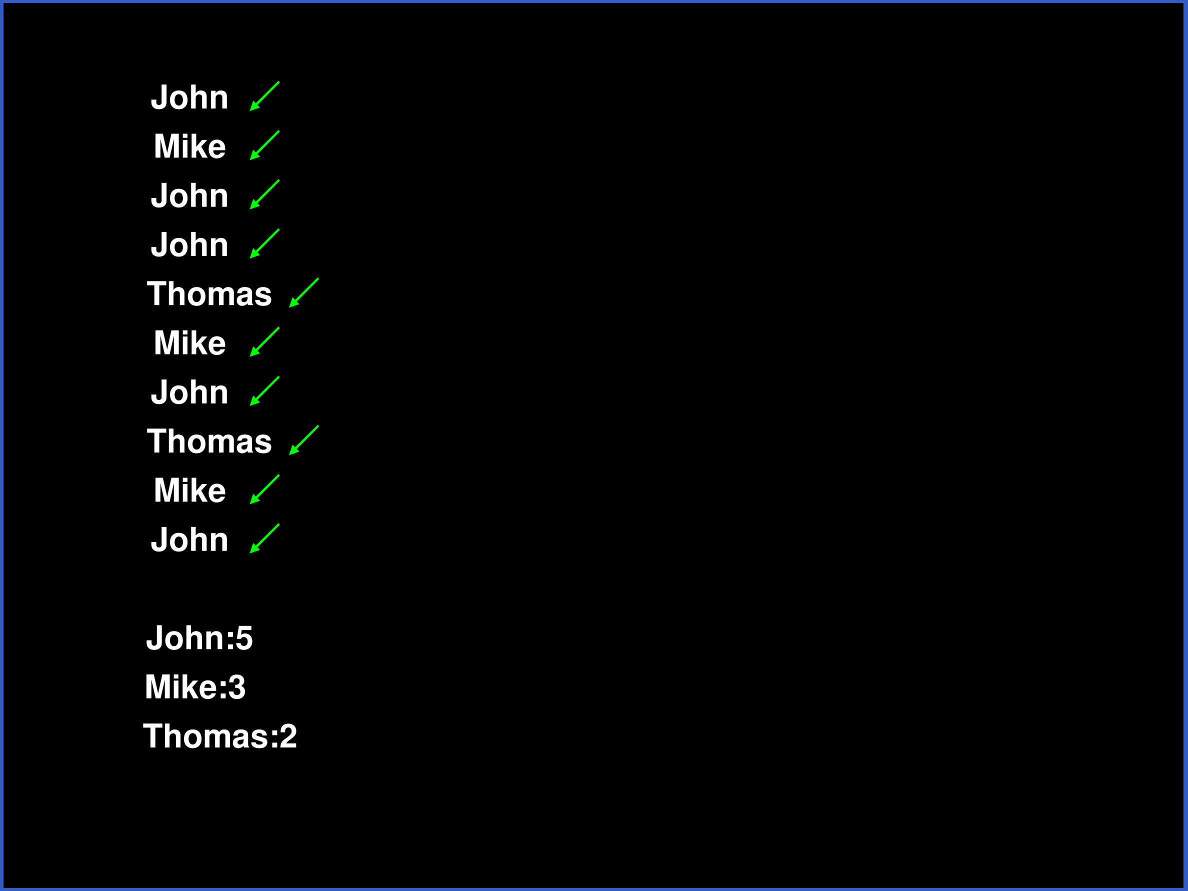
```
# include <iostream>
using namespace std;

struct  Candidate
{
        char  name[20];
        int  count;
} ;

int  main ( )
{
        Candidate leader[3] = {{“John”, 0 }, {“Mike”, 0}, {“Thomas”, 0} };
        char candidate_name[20];
        for ( int i = 0; i < 10; i++ )
        {
                cin >> candidate_name;
                for ( int j = 0; j < 3; j++ )
                      if ( strcmp (candidate_name, leader[ j ].name) == 0)
                                   leader[ j ].count ++ ;
        }
        cout << endl ;

        for ( int i = 0 ; i < 3 ; i++)
            cout << leader[ i ].name << “:” << leader[ i ].count << endl;
}
```

Visual representation of struct leader



	name [20]	count
Leader [0]	**John**	**0**
Leader [1]	**Mike**	**0**
Leader [2]	**Thomas**	**0**

Example 1: Application



```
John ↙
Mike ↙
John ↙
John ↙
Thomas ↙
Mike ↙
John ↙
Thomas ↙
Mike ↙
John ↙

John:5
Mike:3
Thomas:2
```

strcmp function

strcmp. Function strcmp compare its first string argument with its second string argument character by character.

- the 1st string = the 2nd string, return 0
- the 1st string < the 2nd string, return negative value
- the 1st string > the 2nd string, return positive value

strcmp function

strcmp. Function strcmp compare its first string argument with its second string argument character by character.

- the 1st string = the 2nd string, return 0
- the 1st string < the 2nd string, return negative value
- the 1st string > the 2nd string, return positive value

e.g.

- "Boy" > "Axle"
- "Happy Holiday" < "Happy New Year"

Example 2: array of struct data types

Array of `struct` data types II

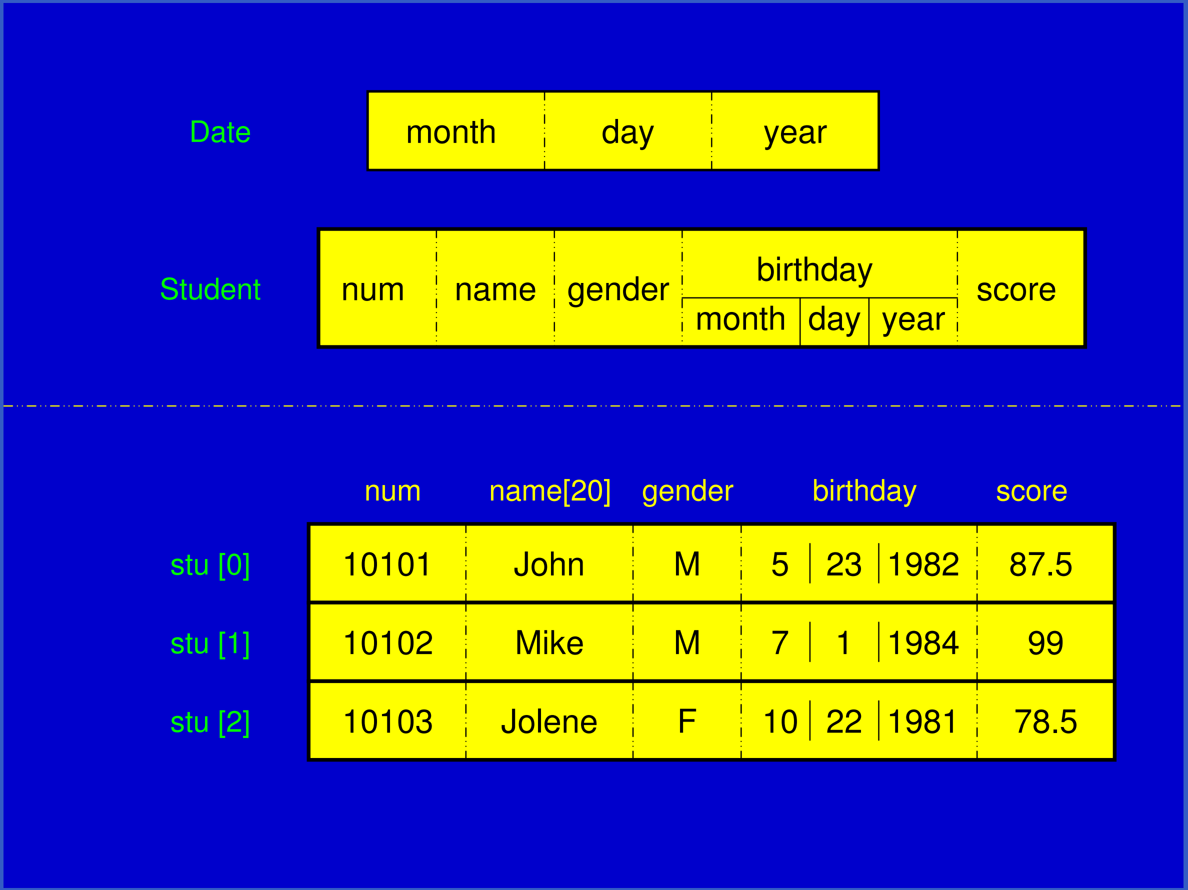
```
# include <iostream>
using namespace std;

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

struct  Student
{
        int  num;                 // student ID number
        char  name[20];
        char  gender;
        Date birthday;
        double  score;
 }  stu[3] = {{10101, "John", 'M', 5, 23, 1982, 87.5},
                  {10102, "Mike", 'M', 7, 1, 1984, 99},
                 {10103, "Jolene", 'F', 10, 22, 1981, 78.5}};
```

Example 2: array of struct data types

Array of `struct` data types II

```
 int  main()
{
       cout << "ID No.\tname\tgender\tbirthday\tscore\n";

       for ( int  i = 0;  i < 3; i++)
       {
              cout << stu[i].num << '\t' << stu[i].name << '\t'
                     << stu[i].gender << '\t';

              cout << stu[i].birthday.month << '/'
                     << stu[i].birthday.day << '/'
                      << stu[i].birthday.year << '\t';

              cout << stu[i].score;

              cout << endl;
        }

       return 0;
 } ;
```

Visual representation of Date & Student



Date

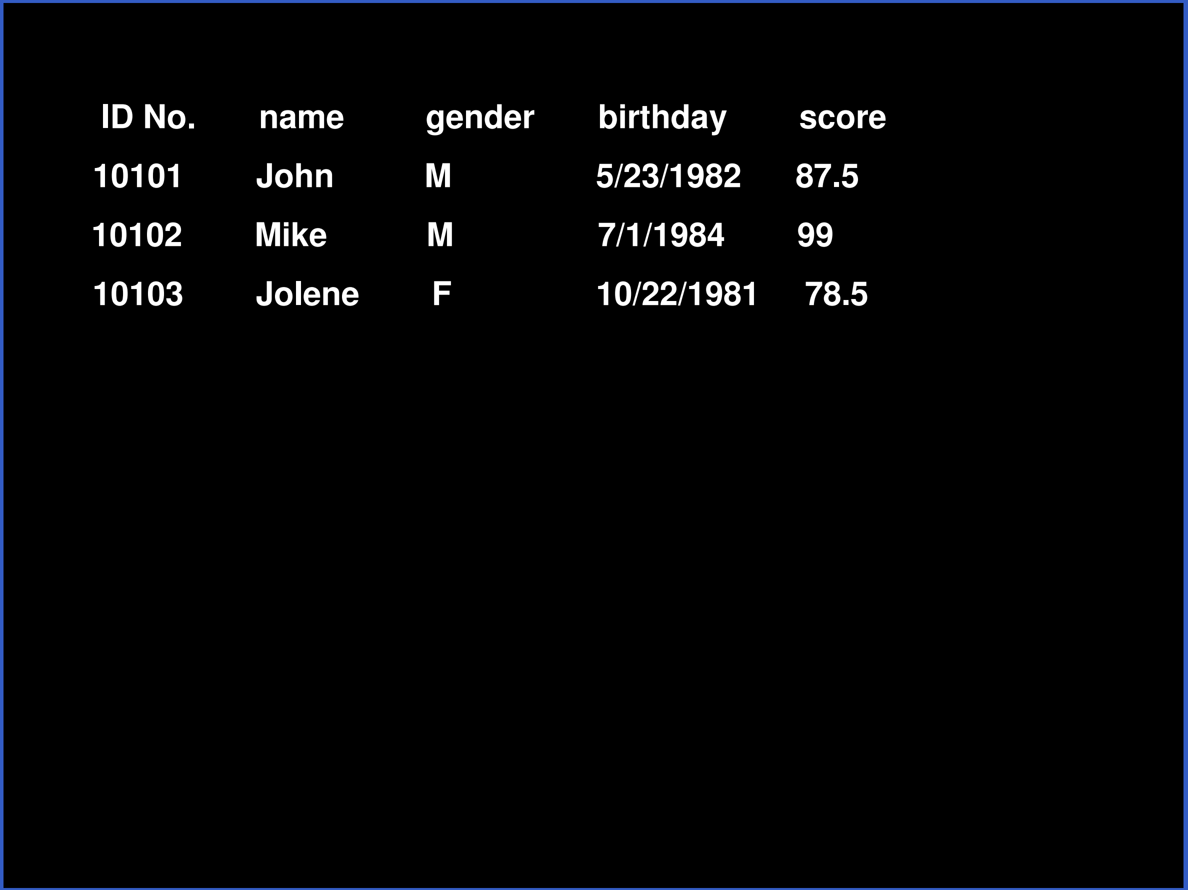
month	day	year

Student

num	name	gender	birthday			score
			month	day	year	

	num	name[20]	gender	birthday			score
stu [0]	10101	John	M	5	23	1982	87.5
stu [1]	10102	Mike	M	7	1	1984	99
stu [2]	10103	Jolene	F	10	22	1981	78.5

Example 2: Application



ID No.	name	gender	birthday	score
10101	John	M	5/23/1982	87.5
10102	Mike	M	7/1/1984	99
10103	Jolene	F	10/22/1981	78.5

Passing struct data to functions

Passing arguments by value and by reference

Recall. Passing arguments

- Pass-by-value. a copy of the arguments'value is made and passed to the called function.

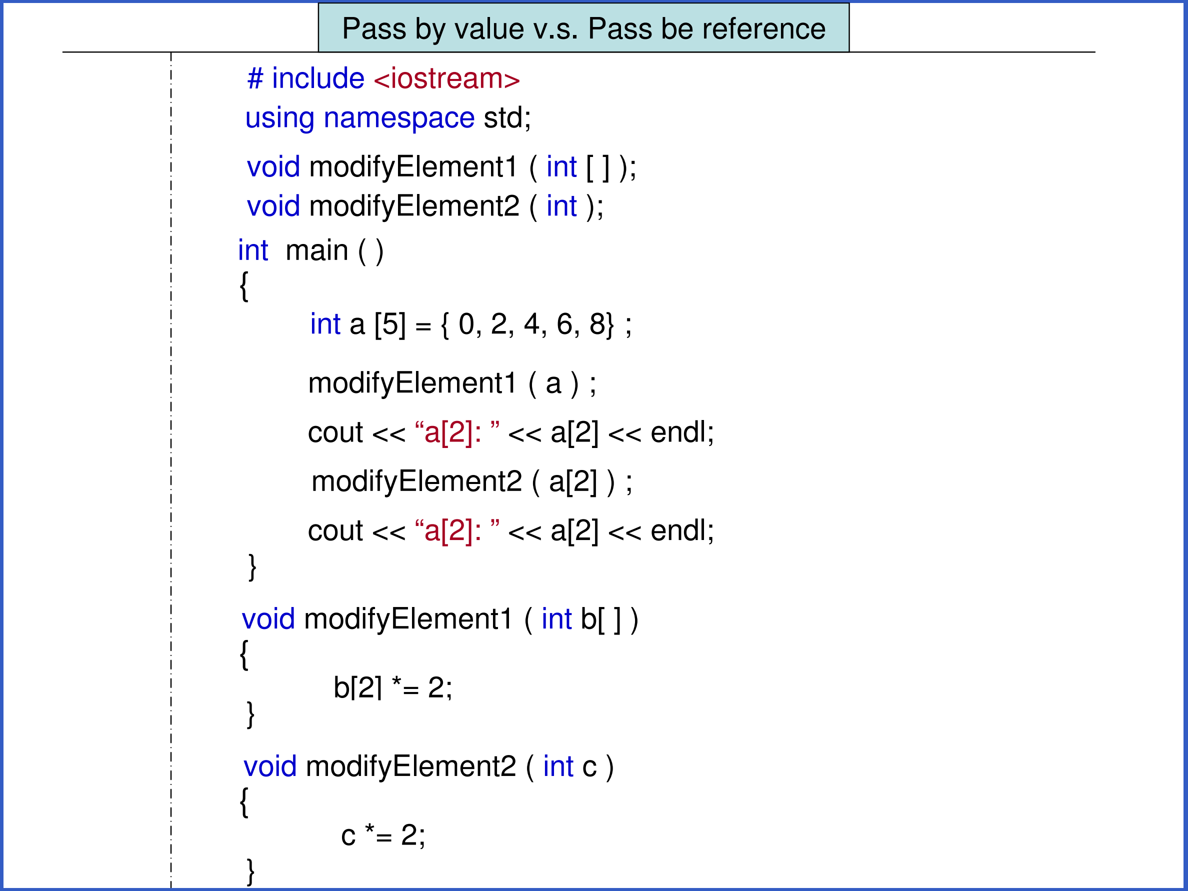
Passing arguments by value and by reference

Recall. Passing arguments

- Pass-by-value. a copy of the arguments'value is made and passed to the called function.

- Pass-by-reference. the caller gives the called function the ability to access the caller's data directly, and to modify the data if the called function chooses to do so.
  — using reference
  — using pointers

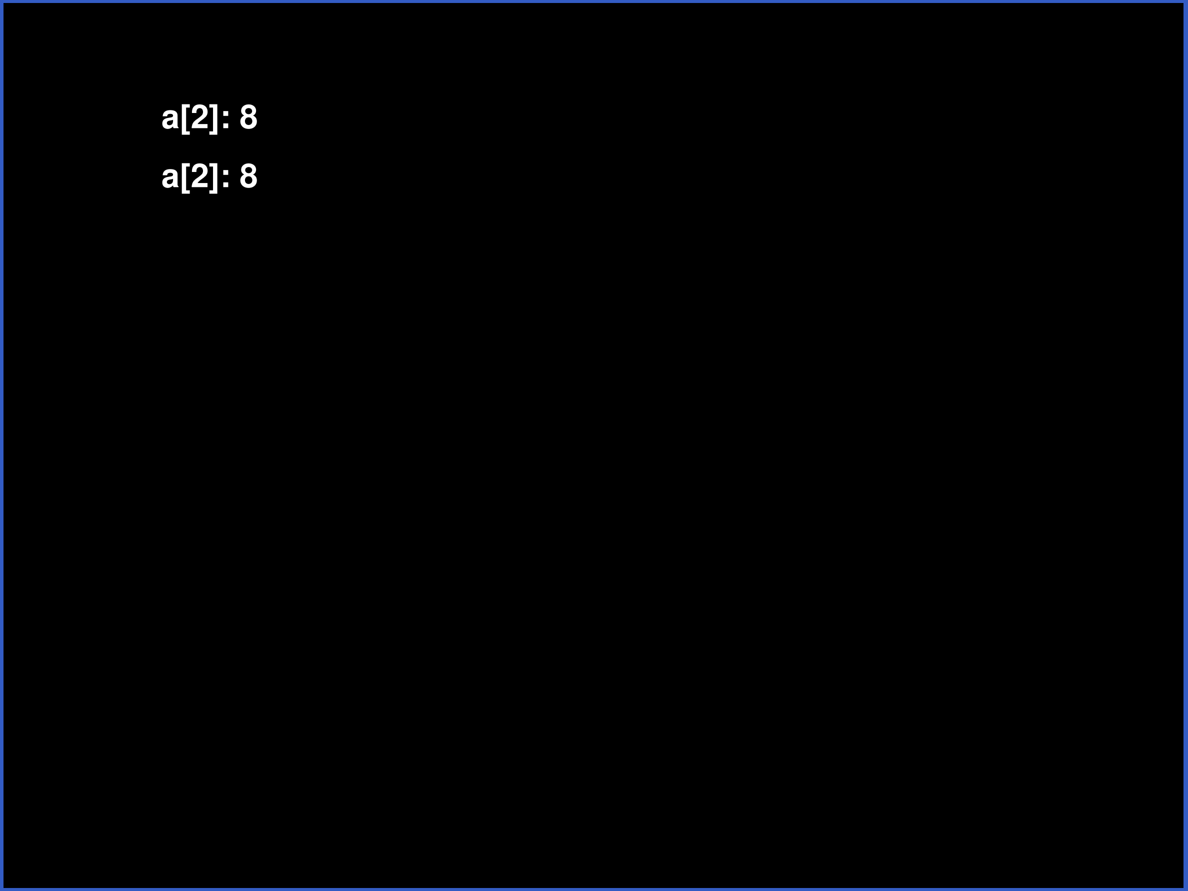
Passing an array to functions



Pass by value v.s. Pass be reference

```
# include <iostream>
using namespace std;
void modifyElement1 ( int [ ] );
void modifyElement2 ( int );
int  main ( )
{
        int a [5] = { 0, 2, 4, 6, 8} ;
        modifyElement1 ( a ) ;
        cout << "a[2]: " << a[2] << endl;
        modifyElement2 ( a[2] ) ;
        cout << "a[2]: " << a[2] << endl;
 }
void modifyElement1 ( int b[ ] )
{
          b[2] *= 2;
 }
void modifyElement2 ( int c )
{
          c *= 2;
 }
```

Output: Passing an array to functions



```
a[2]: 8
a[2]: 8
```

Passing struct data by value and by reference

Facts.

- The entire structure or individual members of a structure can be
  - Passed by value. a copy of the structure is made and passed to the called function.
  - Passed by reference. The address of the structure object or a reference to the structure object would be passed

Passing struct data by value and by reference

Facts.

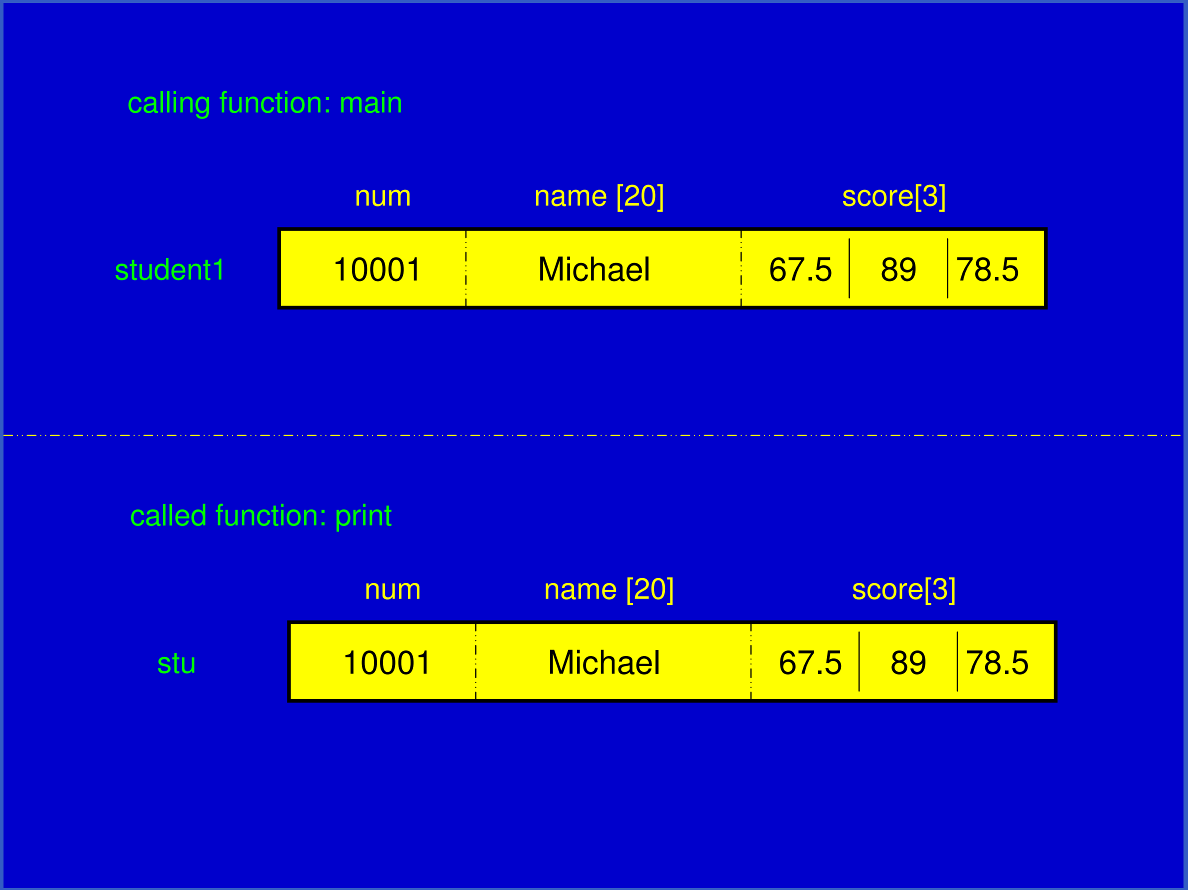
- The entire structure or individual members of a structure can be
  - Passed by value. a copy of the structure is made and passed to the called function.
  - Passed by reference. The address of the structure object or a reference to the structure object would be passed
- By default, structures are passed by value; Passing structures by reference would be more efficient.

Passing struct data by value

Example 1

```
# include <iostream>
using namespace std;

struct  Student
{
        int  num;
        char name[20];
        double  score[3];
} student1 = {10001, “Michael”, 67.5, 89, 78.5} ;

void print ( Student  );

int  main ( )
{
        print ( student1 );
}

void print ( Student  stu )
{
        cout << stu.num << ‘  ’ << stu.name << ‘  ’ << stu.score[0] << ‘  ’
             << stu.score[1] << ‘  ’ << stu.score[2] << ‘  ’ << endl;
}
```

Visual representation: pass-by-value



pass-by-reference using references

Example 2

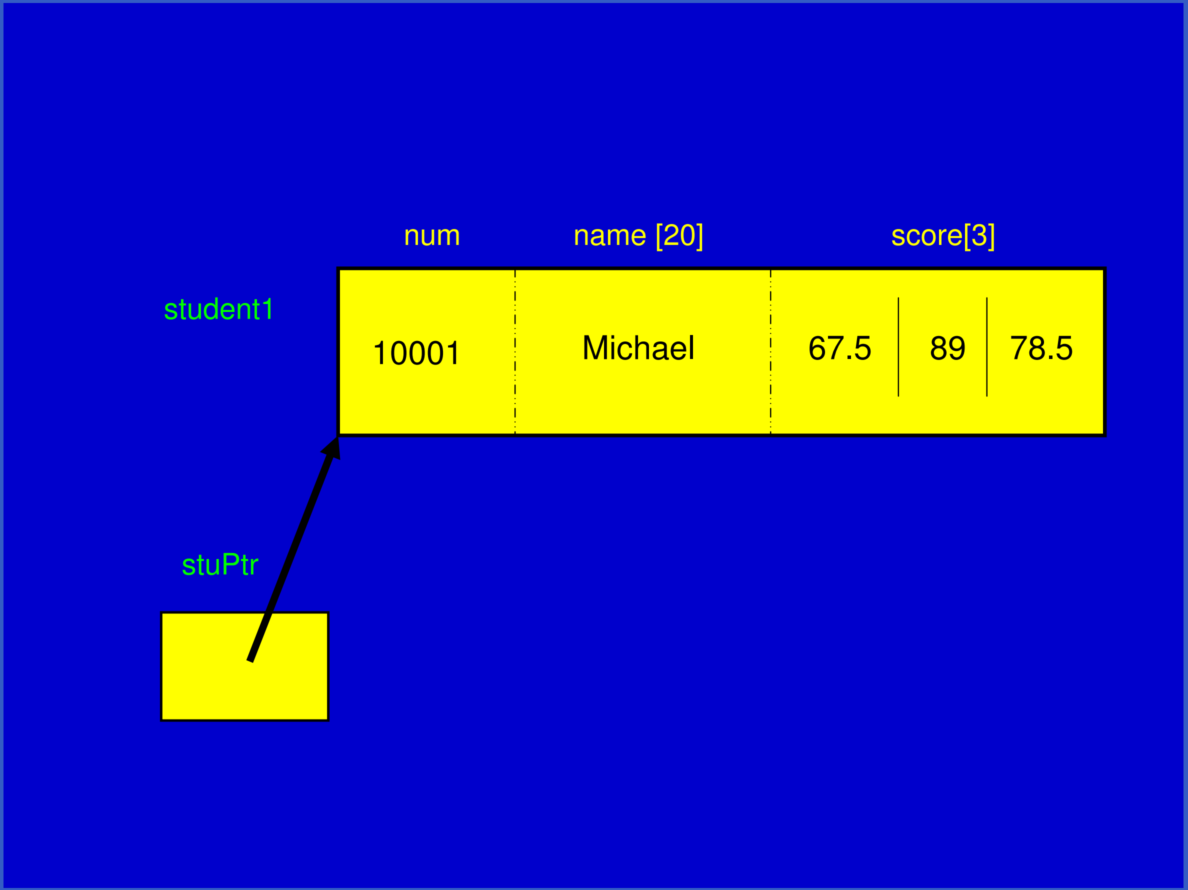
```
# include <iostream>
using namespace std;

struct  Student
{
        int  num;
        char name[20];
        double  score[3];
} student1 = {10001, “Michael”, 67.5, 89, 78.5} ;

void print ( Student  & );

int  main ( )
{
        print ( student1 );           // pass the argument by reference
}

void print ( Student  & stu )
{
        cout << stu.num << ‘  ’ << stu.name << ‘  ’ << stu.score[0] << ‘  ’
             << stu.score[1] << ‘  ’ << stu.score[2] << ‘  ’ << endl;
}
```

pass-by-reference using pointers

Example 3

```
# include <iostream>
using namespace std;

struct  Student
{
     int  num;
     char name[20];
     double  score[3];
} student1 = {10001, “Michael”, 67.5, 89, 78.5} ;

void print ( Student  * );

int  main ( )
{
     print ( & student1 );       // pass the argument by reference
}

void print ( Student  * stuPtr )
{
    cout << (*stuPtr).num << ‘  ’ << (*stuPtr).name << ‘  ’ << (*stuPtr).score[0] << ‘  ’
             << (*stuPtr).score[1] << ‘  ’ << (*stuPtr).score[2] << ‘  ’ << endl;
}
```

Visual representation: pointer stuPtr



pass-by-reference using pointers

Example 3

```
# include <iostream>
using namespace std;

struct  Student
{
     int  num;
     char name[20];
     double  score[3];
} student1 = {10001, “Michael”, 67.5, 89, 78.5} ;

void print ( Student  * );

int  main ( )
{
     print ( & student1 );       // pass the argument by reference
}

void print ( Student  * stuPtr )
{
   cout << stuPtr -> num << ‘  ’ << stuPtr ->name << ‘  ’ << stuPtr -> score[0] << ‘  ’
           << stuPtr -> score[1] << ‘  ’ << stuPtr -> score[2] << ‘  ’ << endl;
}
```