Object Oriented Programming

Shaobai Kan

chapter 7 (Continue...)

Outline

Outline

- Selection sort using pass-by-reference

Outline

- Selection sort using pass-by-reference
- Pointer expressions and pointer arithmetic

Outline

- Selection sort using pass-by-reference
- Pointer expressions and pointer arithmetic
- Relationship between pointers and arrays

Selection sort using pass-by-reference

Selection sort

Fact. Selection sort algorithm is an easy - to - program, but unfortunately inefficient, sorting algorithm.

Selection sort

Fact. Selection sort algorithm is an easy - to - program, but unfortunately inefficient, sorting algorithm.

Selection sort algorithm.

$$24,\ 12,\ 36,\ 5,\ 10 \quad \rightarrow \quad 5,\ 10,\ 12,\ 24,\ 36$$

Selection sort algorithm

Selection sort algorithm

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

void SelectionSort (int * const, const int);
void Swap (int * const, int * const );

int main( )
{
        const int arraySize = 5;
        int a [arraySize] = { 24, 12, 36, 5, 10 };

        SelectionSort (a, arraySize);

        cout << "Data items in ascending order\n" ;

        for (int i = 0; i < arraySize ; i++)
                cout << "\t" <<  a[ i ] ;

        return 0 ;
}

// continue in the next page
```

Selection sort algorithm...

Selection sort algorithm

```
void SelectionSort (int * const array, const int size)
{
        int smallest ;
        for (int i =0 ; i < size ; i++)
        {
              smallest =  i ;

             for (int j =i+1 ; j < size ; j++)
                 if ( array [ j ] < array [ smallest] )
                          smallest = j;

          Swap (&array[ i ], & array [smallest] );
        }
}


 void Swap (int * const p1, int * const p2)
{
     int temp = * p1;
     *p1 = * p2;
     *p2 = temp;
 }
```

Exercise

Exercise 1. Write a program that rearranges the following data in ascending order(using selection sort algorithm).
Data:

13, 24, 100, 9, 56
77, 34, 89, 98, 65
47, 58, 33, 28, 45
18, 14, 16, 52, 49

Pointer expressions and pointer arithmetic

Pointer arithmetic

Fact. C++ enable pointer arithmetic — certain arithmetic operations may be performed on pointers.

- increment (++) / decrement (- -)
- add integer(+, +=) / subtract integer (-, -=)
- one pointer subtracted from another pointer of the same type

Example: pointer arithmetic

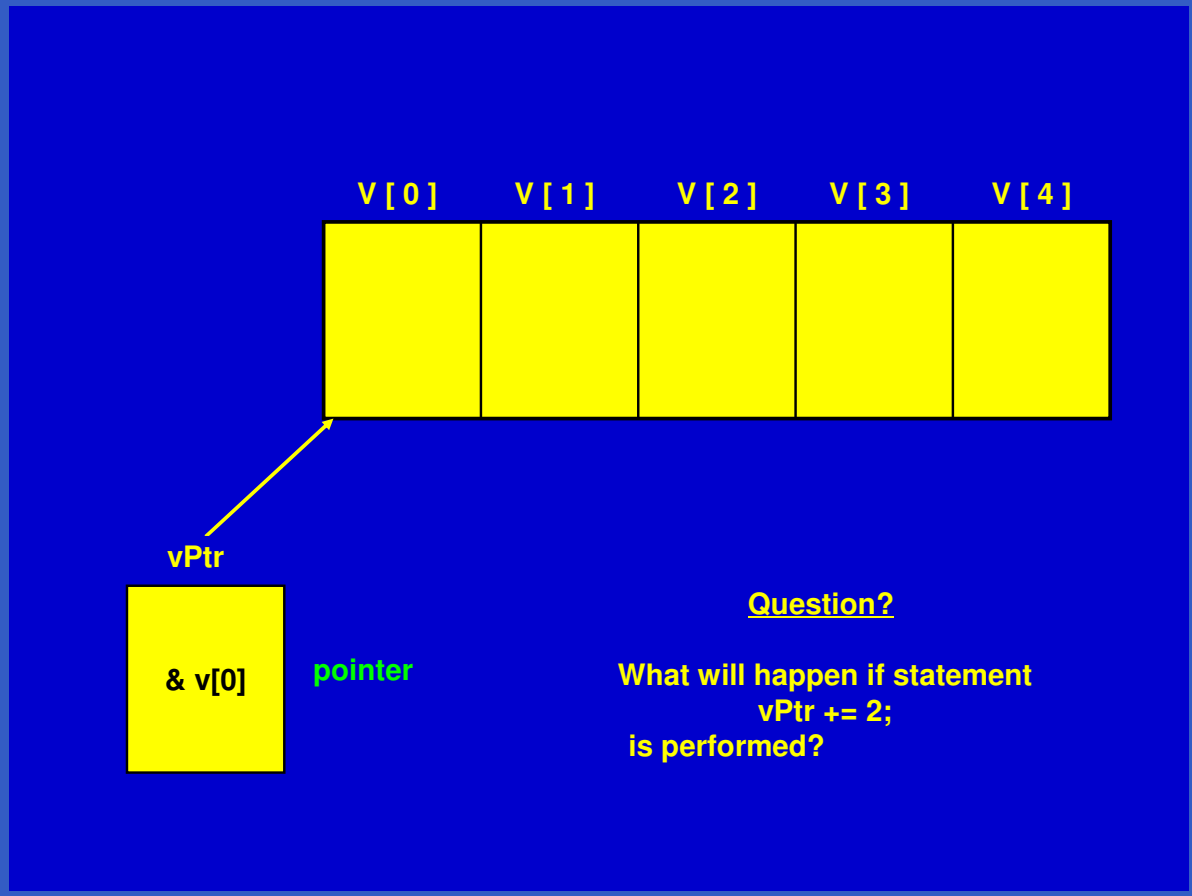
Recall. An array name is just a constant pointer to the beginning of the array.

Example.

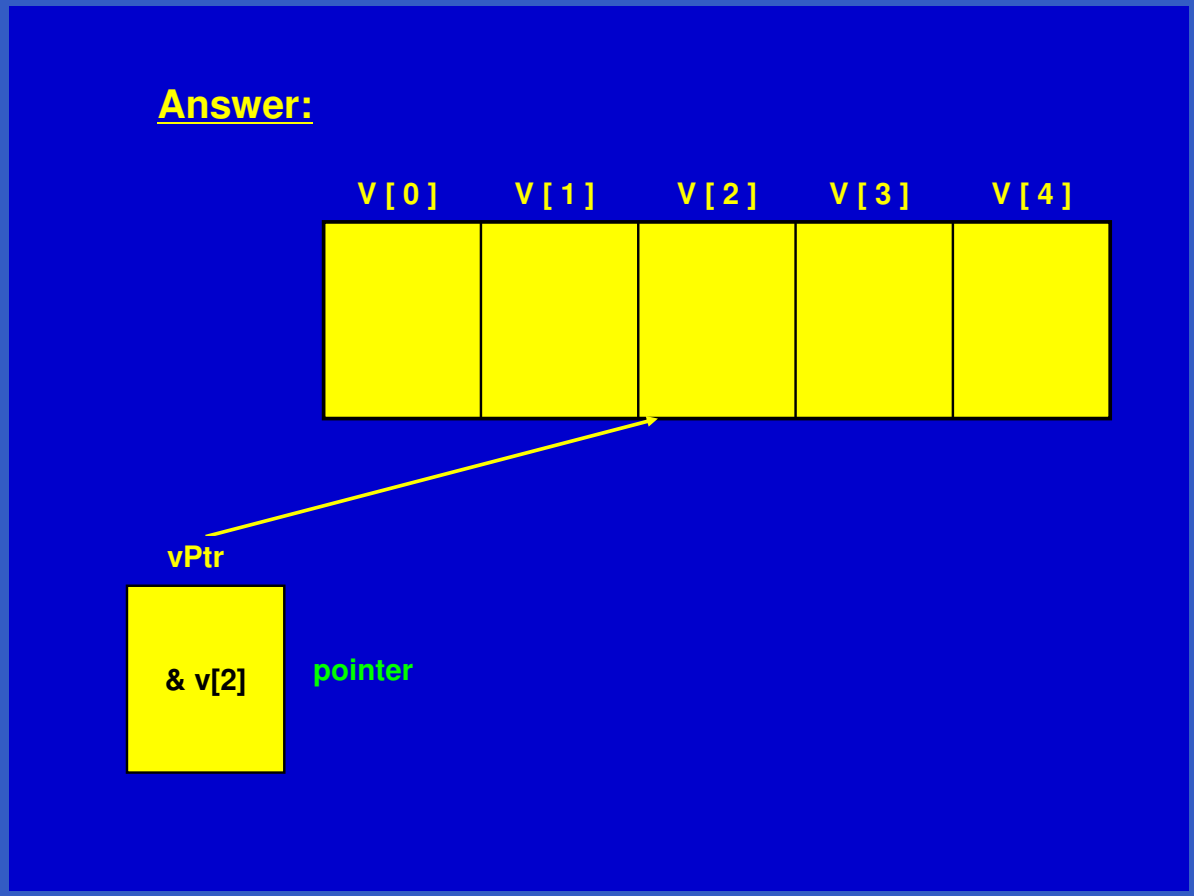
```
int v[5];
int *vPtr = &v[0];     (1)
int *vPtr = v;     (2)
```

Statements (1) and (2) are equivalent.

Example: pointer arithmetic



Example: pointer arithmetic



Pointer Expressions

Questions. What about the following expressions?

- vPtr ++, ++ vPtr.
- vPtr - -, - - vPtr.
- vPtr += 3, vPtr -= 3.

Pointer Expressions

Questions. What about the following expressions?

- vPtr ++, ++ vPtr.
- vPtr - -, - - vPtr.
- vPtr += 3, vPtr -= 3.

Fact. Pointer arithmetic is meaningless unless performed on a pointer that points to an array.

Relationship between pointers and arrays

Pointers *v.s.* arrays

Fact. There are three different notations

- array subscript notation
- pointer subscript notation
- pointer / offset notation

Example

Pointer and array

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

int main( )
{
        int b[ ] = {10, 20 ,30, 40};
        int *bPtr = b;

        for (int i = 0; i < 4; i++)
              cout << "b[" << i << "]:" << b[ i ] << endl;

        for (int i = 0; i < 4; i++)
              cout << "bPtr[" << i << "]:" << bPtr[ i ] << endl;

        for (int i = 0; i < 4; i++)
              cout << "* (b+" << i << "):" << *(b + i) << endl;


         return 0 ;
 }
```

Output

```
b[0]:10
b[1]:20
b[2]:30
b[3]:40
bPtr[0]:10
bPtr[1]:20
bPtr[2]:30
bPtr[3]:40
*(b+0):10
*(b+1):20
*(b+2):30
*(b+3):40
```

Homework:

- Read Sec. 7.6 - 7.9
- practice the program in Fig 7.13 (in the textbook section 7.6)
- practice the program in Fig 7.18 (in the textbook section 7.9)
- Exercise (in this slide)
- Exercise 7.6