ĐẠI NAM
UNIVERSITY

Phần trình bày của:

ĐẬU HẢI PHONG

Giảng viên

Đại Nam, ngày 19 tháng 01 năm 2023

LƯU Ý

KHÔNG NÓI CHUYỆN RIÊNG

KHÔNG SỬ DỤNG ĐIỆN THOẠI

KHÔNG NGỦ GẬT

GHI CHÉP ĐẦY ĐỦ

ĐẠI NAM
UNIVERSITY

LẬP TRÌNH CƠ BẢN

http://dainam.edu.vn

CHƯƠNG 7

TÌM KIẾM VÀ SẮP XẾP

NỘI DUNG

- Tìm kiếm tuyến tính
- Tìm kiếm nhị phân
- Một số thuật toán sắp xếp
- Bài tập

Thuật toán tìm kiếm

- Tìm kiếm: tìm một phần tử trong danh sách các phần tử.
- Xem xét 2 thuật toán tìm kiếm:
  - Tìm kiếm tuyến tính
  - Tìm kiếm nhị phân
- Tìm kiếm tuyến tính:
  - Cách gọi khác: tìm kiếm tuần tự
  - Bắt đầu từ phần tử đầu tiên, thuật toán sẽ lần lượt duyệt và kiểm tra xem từng phần tử trong mảng đến khi tìm được giá trị tìm kiếm.
  - VD: mảng numlist như sau:

17	23	5	11	2	29	3

  - Tìm giá trị 11, thuật toán sẽ kiểm tra 17, 23, 5 và 11.
  - Tìm kiếm 7, thuật toán sẽ kiểm tra 17,23,5,11,2,29,3

Tìm kiếm tuyến tính

- Thuật toán:
  - TimThay = false;
  - ViTri = -1; //Không tìm thấy sẽ nhận giá trị -1, khác với chỉ số mảng
  - ChiSo = 0;
  - While ChiSo <= số lượng phần tử và TimThay = false
    - Nếu List[ChiSo] = GiaTriTimKiem:
      - TimThay = True;
      - ViTri = ChiSo;
    - ChiSo++;
  - Trả về ViTri;

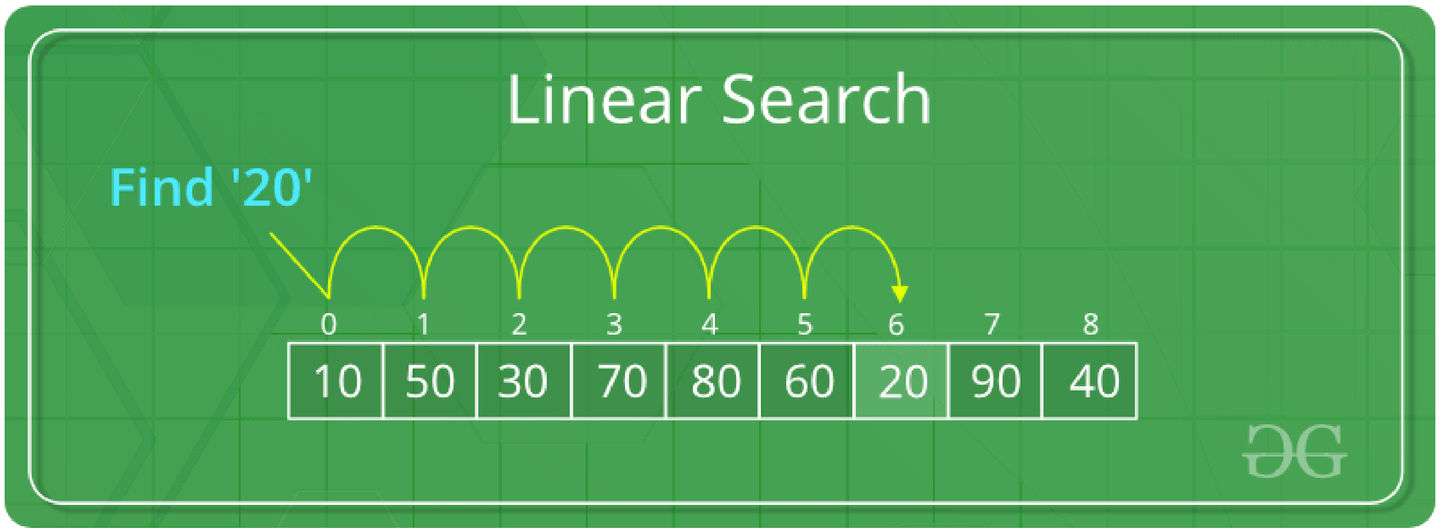
Hàm tìm kiếm tuyến tính

```
int searchList(int list[], int numElems, int value)
{
   int index = 0;        // Used as a subscript to search array
   int position = -1;    // To record position of search value
   bool found = false; // Flag to indicate if value was found

   while (index < numElems && !found)
   {
      if (list[index] == value) // If the value is found
      {
         found = true; // Set the flag
         position = index; // Record the value's subscript
      }
      index++; // Go to the next element
   }
return position; // Return the position, or -1
}
```

Ưu điểm – Nhược điểm

- Ưu điểm:
  - Dễ dàng để hiểu thuật toán
  - Không yêu cầu các phần tử lưu trữ trong mảng theo thứ tự nào.
- Nhược điểm:
  - Chậm vì với mảng gồm N phần tử thì phải kiểm tra trung bình là N/2 nếu phần tử xuất hiện trong mảng và N nếu phần tử không có trong mảng.



Tìm kiếm nhị phân

- Yêu cầu các phần tử trong mảng được sắp xếp theo thứ tự
- B1: Chia mảng thành 3 phần:
  - Chỉ số giữa
  - Các phần tử trong 1 bên chỉ số giữa
  - Các phần tử trong 1 bên còn lại của chỉ số giữa

- B2: Nếu phần tử ở chỉ số giữa là phần tử cần tìm thì XONG. Còn không quay lại B1, chỉ dùng nửa mảng có thể chứa giá trị cần tìm.
- B3: Lặp lại B1 & B2 đến khi tìm được giá trị cần tìm hoặc không tìm được.

Tìm kiếm nhị phân

- Giả sử mảng numlist2 như sau:

2	3	5	11	17	23	29

  - Tìm giá trị 11, thuật toán nhị phân kiểm tra đúng 11, dừng
  - Tìm giá trị 7, thuật toán nhị phân kiểm tra 2, 3, 5 & dừng
- Thuật toán:
  - ChiSoDau = 0;
  - ChiSoCuoi = chỉ số cuối của mảng;
  - TimThay = false; ViTri = -1;
  - While TimThay = false & ChiSoDau <= ChiSoCuoi
    - ChiSoGiua = (ChiSoDau + ChiSoCuoi)/2;
    - Nếu array[ChiSoGiua] = PhanTuTim
      - TimThay = True;
      - ViTri = ChiSoGiua;

Copyright by Dau Hai Phong - 2023

Tìm kiếm nhị phân

- Thuật toán:
  - ChiSoDau = 0;
  - ChiSoCuoi = chỉ số cuối của mảng;
  - TimThay = false; ViTri = -1;
  - While TimThay = false & ChiSoDau <= ChiSoCuoi
    - ChiSoGiua = (ChiSoDau + ChiSoCuoi)/2
    - Nếu Mang[ChiSoGiua] = PhanTuTim
      - TimThay = True;
      - ViTri = ChiSoGiua;
    - Ngược lại, nếu Mang[ChiSoGiua] > PhanTuTim thì:
      - ChiSoCuoi = ChiSoGiua -1;
    - Ngược lại,
      - ChiSoDau = ChiSoGiua + 1;
  - Return ViTri;

Copyright by Dau Hai Phong - 2023
Copyright by Dau Hai Phong - 2023

Tìm kiếm nhị phân

```
int binarySearch(int array[], int size, int value)
{
  int first = 0,            // First array element
  int last = size - 1,      // Last array element
  int middle,               // Mid point of search
  position = -1;        // Position of search value
  bool found = false;       // Flag

  while (!found && first <= last)
  {
    middle = (first + last) / 2;     // Calculate mid point
    if (array[middle] == value)      // If value is found at mid
    {
      found = true;
      position = middle;
    }
    else if (array[middle] > value)  // If value is in lower half
      last = middle - 1;
    else
      first = middle + 1;          // If value is in upper half
  }
  return position;
}
```


Copyright by Dau Hai Phong - 2023

Ưu điểm – Nhược điểm

- Ưu điểm:
  - Hiệu quả hơn tìm kiếm tuyến tính
  - Với mảng gồm N phần tử, đa số thực hiện với $\log_2 N$
- Nhược điểm:
  - Yêu cầu các phần tử mảng phải được sắp xếp

Kiểm tra

- 7.1. Miêu tả sự khác nhau giữa tìm kiếm tuyến tính và tìm kiếm nhị phân?
- 7.2. Với 1 mảng gồm 20.000 phần tử, trung bình bao nhiêu so sánh để tìm kiếm tuyến tính?(giả sử có xuất hiện trong mảng).
- 7.3. Với 1 mảng gồm 20.000 phần tử, số lần so sánh tối đa để tìm kiếm nhị phân?
- 7.4. Một tìm kiếm tuyến tính được thực hiện trên 1 mảng, biết rằng một số phần tử thường xuyên được tìm hơn các phần tử khác. Vậy các tổ chức lưu trữ của mảng nên như thế nào để cải thiện tốc độ tìm kiếm trung bình?

Copyright by Dau Hai Phong - 2023
Copyright by Dau Hai Phong - 2023

Bài tập thực tế

- Trung tâm quản lý thông tin về sách, DVDs, CDs âm thanh như sau:

Table 8-1 Products

Product Title	Product Description	Product Number	Unit Price
Six Steps to Leadership	Book	914	$12.95
Six Steps to Leadership	Audio CD	915	$14.95
The Road to Excellence	DVD	916	$18.95
Seven Lessons of Quality	Book	917	$16.95
Seven Lessons of Quality	Audio CD	918	$21.95
Seven Lessons of Quality	DVD	919	$31.95
Teams Are Made, Not Born	Book	920	$14.95
Leadership for the Future	Book	921	$14.95
Leadership for the Future	Audio CD	922	$16.95

  - Người quản lý yêu cầu viết chương trình tìm kiếm sản phẩm. Chương trình cho phép người dùng nhập vào mã sản phẩm, sau đó sẽ hiển thị tiêu đề, mô tả và giá của sản phẩm đó.

Bài tập thực tế

Table 8-2 Variables

Variable	Description
`NUM_PRODS`	A constant integer initialized with the number of products the Demetris Leadership Center sells. This value will be used in the definition of the program's array.
`MIN_PRODNUM`	A constant integer initialized with the lowest product number.
`MAX_PRODNUM`	A constant integer initialized with the highest product number.
`id`	Array of integers. Holds each product's number.
`title`	Array of strings, initialized with the titles of products.
`description`	Array of strings, initialized with the descriptions of each product.
`prices`	Array of `doubles`. Holds each product's price.

Modules

The program will consist of the functions listed in Table 8-3.

Table 8-3 Functions

Function	Description
`main`	The program's `main` function. It calls the program's other functions.
`getProdNum`	Prompts the user to enter a product number. The function validates input and rejects any value outside the range of correct product numbers.
`binarySearch`	A standard binary search routine. Searches an array for a specified value. If the value is found, its subscript is returned. If the value is not found, –1 is returned.
`displayProd`	Uses a common subscript into the `title`, `description`, and `prices` arrays to display the title, description, and price of a product.

Copyright by Dau Hai Phong - 2021

Bài tập thực tế

- Hàm main

```
do
    Call getProdNum
    Call binarySearch
    If binarySearch returned -1
        Inform the user that the product number was not found
    else
        Call displayProd
    End If
    Ask the user if the program should repeat
While the user wants to repeat the program
```

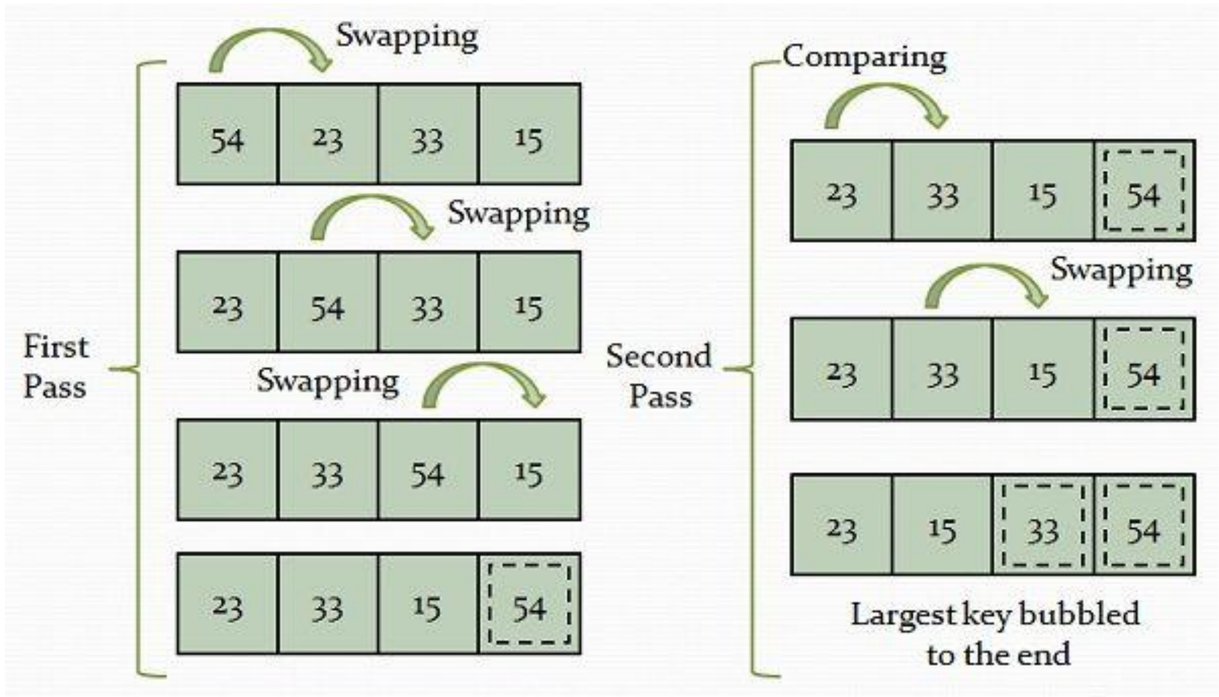
```
Display a prompt to enter a product number
Read prodNum
While prodNum is invalid
    Display an error message
    Read prodNum
End While
Return prodNum
```

```
void displayProd(const string title[], const string desc[],
                 const double price[], int index)
{
    cout << "Title: " << title[index] << endl;
    cout << "Description: " << desc[index] << endl;
    cout << "Price: $" << price[index] << endl;
}
```

Copyright by Dau Hai Phong - 2023
Copyright by Dau Hai Phong - 2023

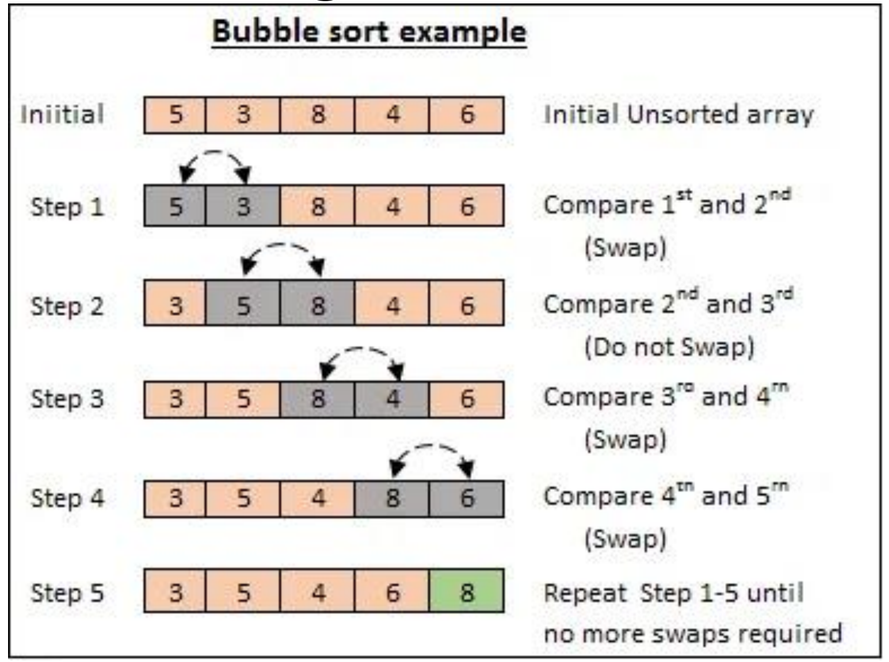
Thuật toán sắp xếp

- Sắp xếp: bố trí các giá trị theo thứ tự:
  - Bảng chữ cái
  - Số tăng dần
  - Số giảm dần
- Hai thuật toán cần quan tâm:
  - Sắp xếp nổi bọt (Bubble sort)
  - Sắp xếp lựa chọn (Selection sort)



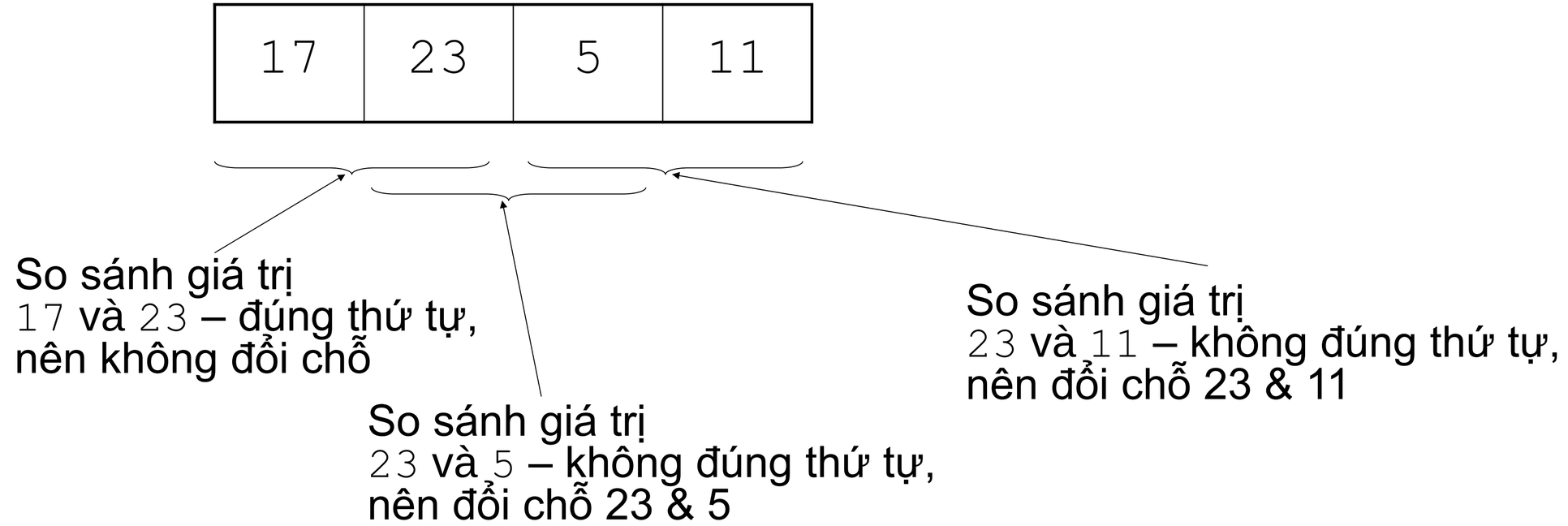
Sắp xếp nổi bọt - Bubble sort

- Ý tưởng:
  - So sánh cặp phần tử đầu tiên,
    - Nếu không đúng thứ tự thì đổi chỗ cặp phần tử
  - Di chuyển xuống một phần tử, so sánh phần tử thứ 2 và thứ 3, đổi chỗ nếu chưa đúng vị trí. Tiếp tục cho đến khi hết mảng.
  - Duyệt lại các phần tử mảng, đổi chỗ nếu chưa đúng vị trí.
  - Lặp lại việc duyệt mảng đến khi không có còn phải đổi chỗ.



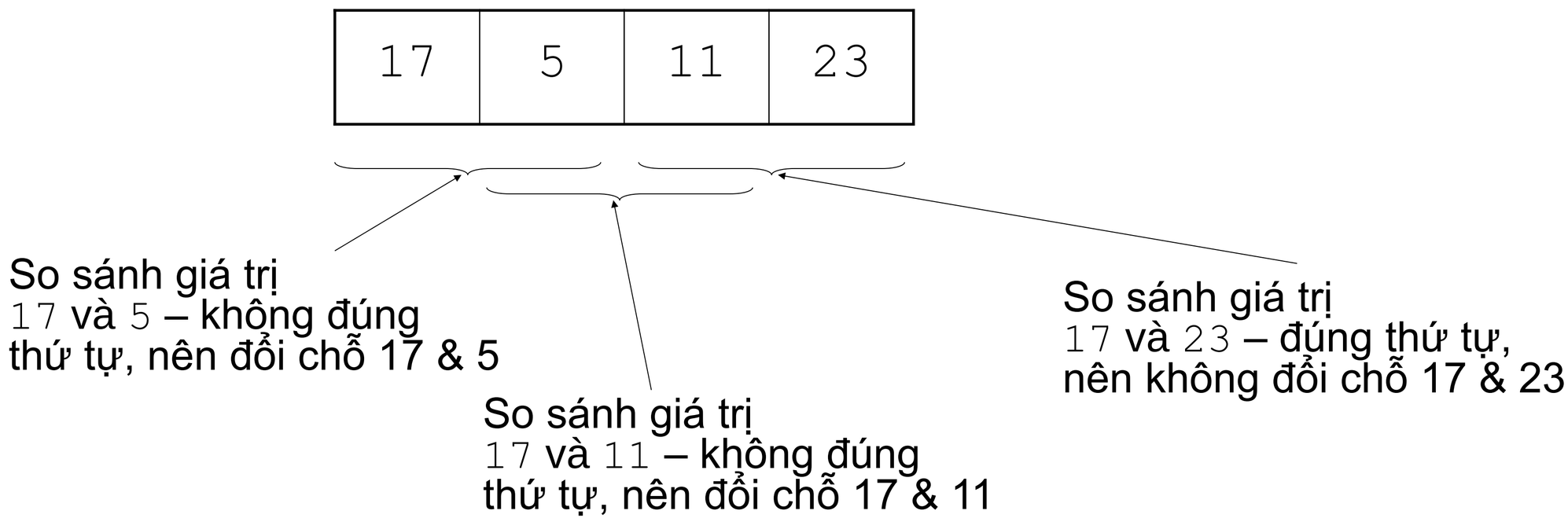
Sắp xếp nổi bọt - Bubble sort

- Giả sử mảng numlist3 bao gồm:
  - Lần duyệt 1:



Sắp xếp nổi bọt - Bubble sort

- Sau lần duyệt 1, mảng numlist3 bao gồm:
  - Lần duyệt 2:



Sắp xếp nổi bọt - Bubble sort

- Sau lần duyệt 2, mảng numlist3 bao gồm:
  - Lần duyệt 3:

5	11	17	23

So sánh giá trị
5 và 11 – đúng thứ tự,
nên không đổi chỗ

So sánh giá trị
11 và 17 – đúng thứ tự,
nên không đổi chỗ

So sánh giá trị
17 và 23 – đúng thứ tự,
nên không đổi chỗ

Không có sự đổi chỗ,
nên mảng đã được sắp xếp

Hàm sắp xếp nổi bọt

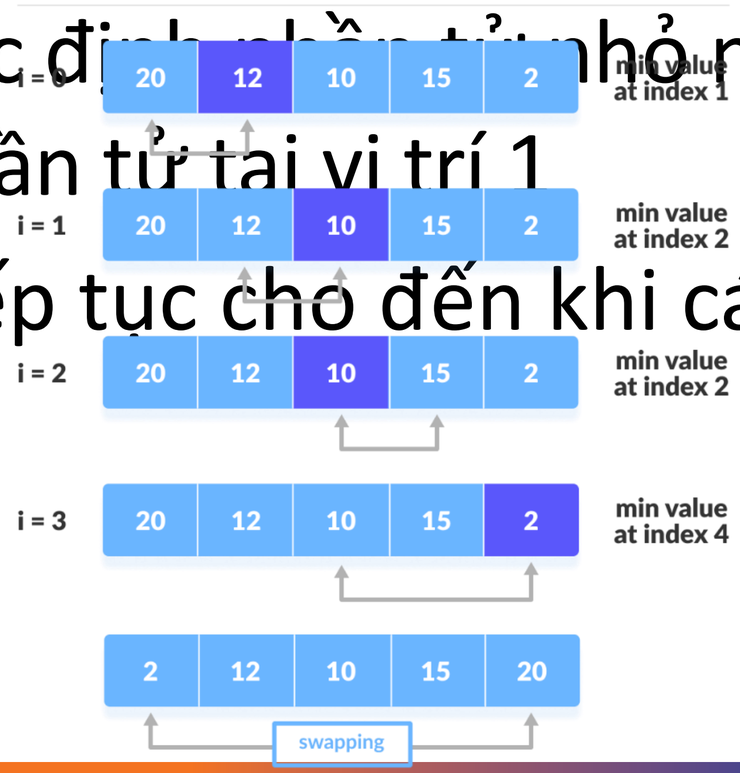
```
34 void sortArray(int array[], int size)
35 {
36    bool swap;
37    int temp;
38
39    do
40    {
41       swap = false;
42       for (int count = 0; count < (size - 1); count++)
43       {
44          if (array[count] > array[count + 1])
45          {
46             temp = array[count];
47             array[count] = array[count + 1];
48             array[count + 1] = temp;
49             swap = true;
50          }
51       }
52    } while (swap);
53 }
```

Ưu điểm – Nhược điểm

- Ưu điểm:
  - Dễ hiểu & cài đặt
- Nhược điểm:
  - Không hiệu g lớn

Sắp xếp lựa chọn – Selection sort

- Ý tưởng: giả sử theo thứ tự tăng dần
  - Xác định phần tử nhỏ nhất trong mảng. Đổi chỗ với phần tử tại vị trí 0
  - Xác định phần tử nhỏ nhất trong mảng. Đổi chỗ với phần tử tại vị trí 1
  - Tiếp tục cho đến khi các phần tử trong mảng theo thứ tự



Sắp xếp lựa chọn – Selection sort

- Ví dụ: mảng numlist gồm:

11	2	29	3

- Lần 1: phần tử nhỏ nhất là 2, đổi chỗ 2 với phần tử có vị trí đầu tiên trong mảng.

2	11	29	3

- Lần 2: phần tử nhỏ tiếp theo là 3, đổi chỗ 3 với phần tử có vị trí thứ 2 trong mảng

2	3	29	11

- Lần 3: phần tử nhỏ tiếp theo là 11, đổi chỗ 11 với phần tử có vị trí thứ 3 trong mảng

2	3	11	29

Hàm sắp xếp lựa chọn

```
35 void selectionSort(int array[], int size)
36 {
37    int startScan, minIndex, minValue;
38
39    for (startScan = 0; startScan < (size - 1); startScan++)
40    {
41       minIndex = startScan;
42       minValue = array[startScan];
43       for(int index = startScan + 1; index < size; index++)
44       {
45          if (array[index] < minValue)
46          {
47             minValue = array[index];
48             minIndex = index;
49          }
50       }
51       array[minIndex] = array[startScan];
52       array[startScan] = minValue;
53    }
54 }
```

Ưu điểm – Nhược điểm

- Ưu điểm:
  - Hiệu quả hơn sắp xếp nổi bọt vì đổi chỗ ít hơn
- Nhược điểm:
  - Khó hiểu hơ xếp nổi bọt

Sắp xếp và Tìm kiếm trên véc tơ

- Thuật toán Sắp xếp và Tìm kiếm có thể áp dụng trên véc tơ giống như mảng.
- Cần sửa đổi nhẹ trong hàm có sử dụng tham số véc tơ:

Program 8-7

```
1  // This program demonstrates how to sort and search a
2  // string vector using selection sort and binary search.
3  #include <iostream>
4  #include <string>
5  #include <vector>
6  using namespace std;
7
8  // Function prototypes
9  void selectionSort(vector<string>&);
10 void swap(string &, string &);
11 int binarySearch(const vector<string>&, string);
12
```

thể sử dụng
àn tử.

Sắp xếp và Tìm kiếm trên véc tơ

```
13  int main()
14  {
15     string searchValue;  // Value to search for
16     int position;        // Position of found value
17
18     // Define a vector of strings.
19     vector<string> names{ "Lopez", "Smith", "Pike", "Jones",
20                           "Abernathy", "Hall", "Wilson", "Kimura",
21                           "Alvarado", "Harrison", "Geddes", "Irvine" };
22
23     // Sort the vector.
24     selectionSort(names);
25
26     // Display the vector's elements.
27     cout << "Here are the sorted names:\n";
28     for (auto element : names)
29        cout << element << endl;
30     cout << endl;
31
32     // Search for a name.
33     cout << "Enter a name to search for: ";
34     getline(cin, searchValue);
35     position = binarySearch(names, searchValue);
36
37     // Display the results.
38     if (position != -1)
39        cout << "That name is found at position " << position << endl;
40     else
41        cout << "That name is not found.\n";
42
43     return 0;
44  }
45
```

Copyright by Dau Hai Phong - 2023
Copyright by Dau Hai Phong - 2023

Sắp xếp và Tìm kiếm trên véc tơ

```
46  //*************************************************************************
47  // The selectionSort function sorts a string vector in ascending order.*
48  //*************************************************************************
49  void selectionSort(vector<string> &v)
50  {
51     int minIndex;
52     string minValue;
53
54     for (int start = 0; start < (v.size() - 1); start++)
55     {
56        minIndex = start;
57        minValue = v[start];
58        for (int index = start + 1; index < v.size(); index++)
59        {
60           if (v[index] < minValue)
61           {
62              minValue = v[index];
63              minIndex = index;
64           }
65        }
66        swap(v[minIndex], v[start]);
67     }
68  }
69
```

```
70  //*********************************************************
71  // The swap function swaps a and b in memory.      *
72  //*********************************************************
73  void swap(string &a, string &b)
74  {
75     string temp = a;
76     a = b;
77     b = temp;
78  }
79
```


Copyright by Dau Hai Phong - 2023

Sắp xếp và Tìm kiếm trên véc tơ

```
80  //*************************************************************
81  // The binarySearch function performs a binary search on a    *
82  // string vector. array, which has a maximum of size elements,*
83  // is searched for the number stored in value. If the number is*
84  // found, its vector subscript is returned. Otherwise, -1 is  *
85  // returned indicating the value was not in the vector.       *
86  //*************************************************************
87
88  int binarySearch(const vector<string> &v, string str)
89  {
90     int first = 0,              // First vector element
91         last = v.size() - 1,    // Last vector element
92         middle,                 // Mid point of search
93         position = -1;          // Position of search value
94     bool found = false;         // Flag
95
96     while (!found && first <= last)
97     {
98        middle = (first + last) / 2; // Calculate mid point
99        if (v[middle] == str)        // If value is found at mid
100       {
101          found = true;
102          position = middle;
103       }
104       else if (v[middle] > str)    // If value is in lower half
105          last = middle - 1;
106       else
107          first = middle + 1;      // If value is in upper half
108    }
109    return position;
110 }
```

Program Output with Example Input Shown in Bold

```
Here are the sorted names:
Abernathy
Alvarado
Geddes
Hall
Harrison
Irvine
Jones
Kimura
Lopez
Pike
Smith
Wilson

Enter a name to search for: Lopez Enter
That name is found at position 8
```

Program Output with Example Input Shown in Bold

```
Here are the sorted names:
Abernathy
Alvarado
Geddes
Hall
```

Copyright by Dau Hai Phong - 2023
Copyright by Dau Hai Phong - 2023

Bài tập thực tế

- Trung tâm quản lý thông tin về sách, DVDs, CDs âm thanh, lưu trữ số lượng bán của các sản phẩm trong 6 tháng như sau:
- Lãnh đạo yêu cầu viết chương trình báo cáo bán hàng về những thông tin sau:
  - Danh sách mã sản phẩm được sắp xế giảm dần (không gồm số lượng bán).
  - Tổng số lượng đã bán
  - Tổng doanh số đã bán trong 6 tháng.

Gợi ý: Doanh số = Tổng Số lượng * Giá bán

- Số lượng là Unit Sold trong bảng 8-4
- Giá bán là Unit Price trong bảng 8-1 ở tập trước

Table 8-4 Units Sold

Product Number	Units Sold
914	842
915	416
916	127
917	514
918	437
919	269
920	97
921	492
922	212

Bài tập thực tế

Biến	Mô tả
NUM_PRODS	Hằng số nguyên, khởi tạo số lượng sản phẩm
prodNum	Mảng nguyên, lưu trữ mã sản phẩm

Moduls	Mô tả
main	Hàm chính, để gọi các hàm khác
calcSales	Tính toán thành tiền cho mỗi sản phẩm
dualSort	Sắp xếp mảng sales giảm dần nên cũng phải sắp xếp các phần tử trong

Bài tập thực tế

- Hàm main
  - Giả m

```
Call calcSales
Call dualSort
Set display mode to fixed point with two decimal places of precision
Call showOrder
Call showTotals
```

- C++:

```
// Calculate each product's sales.
calcSales(units, prices, sales, NUM_PRODS);
// Sort the elements in the sales array in descending
// order and shuffle the ID numbers in the id array to
// keep them in parallel.
dualSort(id, sales, NUM_PRODS);

// Set the numeric output formatting.
cout << setprecision(2) << fixed << showpoint;

// Display the products and sales amounts.
showOrder(sales, id, NUM_PRODS);

// Display total units sold and total sales.
showTotals(sales, units, NUM_PRODS);
```

Copyright by Dau Hai Phong - 2023
Copyright by Dau Hai Phong - 2023

Bài tập thực tế

- Hàm calcSales:
  - (

```
For index = each array subscript from 0 through the last subscript
    sales[index] = units[index] * prices[index]
End For
```

  - C++

```
void calcSales(const int units[], const double prices[],
               double sales[], int num)
{
    for (int index = 0; index < num; index++)
      sales[index] = units[index] * prices[index];
}
```

Bài tập thực tế

- Hàm dualSort:
  - Giả mã:

```
For start = each array subscript, from the first to the next-to-last
    index = start
    maxIndex = start
    tempId = id[start]
    maxValue = sales[start]
    For index = start + 1 To size - 1
        If sales[index] > maxValue
            maxValue = sales[index]
            tempId = id[index]
            maxIndex = index
        End If
    End For
    swap sales[maxIndex] with sales[start]
    swap id[maxIndex] with id[start]
End For
```

  - C++:

```
void dualSort(int id[], double sales[], int size)
{
    int start, maxIndex, tempid;
    double maxValue;

    for (start = 0; start < (size - 1); start++)
    {
        maxIndex = start;
        maxValue = sales[start];
        tempid = id[start];
        for (int index = start + 1; index < size; index++)
        {
            if (sales[index] > maxValue)
            {
                maxValue = sales[index];
                tempid = id[index];
                maxIndex = index;
            }
        }
        swap(sales[maxIndex], sales[start]);
        swap(id[maxIndex], id[start]);
    }
}
```

Copyright by Dau Hai Phong - 2023
Copyright by Dau Hai Phong - 2023

Bài tập thực tế

- Hàm swap (quá tải):

```
//*********************************************************
// The swap function swaps doubles a and b in memory. *
//*********************************************************
void swap(double &a, double &b)
{
   double temp = a;
   a = b;
   b = temp;
}

//*********************************************************
// The swap function swaps ints a and b in memory.     *
//*********************************************************
void swap(int &a, int &b)
{
   int temp = a;
   a = b;
   b = temp;
}
```

Bài tập thực tế

- Hàm showOrder:
  - Giả

```
Display heading
For index = each subscript of the arrays from 0 through the last subscript
    Display id[index]
    Display sales[index]
End For
```

  - C++

```
void showOrder(const double sales[], const int id[], int num)
{
    cout << "Product Number\tSales\n";

    cout << "-----------------------------------\n";
    for (int index = 0; index < num; index++)
    {
        cout << id[index] << "\t\t$";
        cout << setw(8) << sales[index] << endl;
    }
    cout << endl;
}
```

Bài tập thực tế

- Hàm showTotal:
  - Giả m

```
totalUnits = 0
totalSales = 0.0
For index = each array subscript from 0 through the last subscript
    Add units[index] to totalUnits[index]
    Add sales[index] to totalSales
End For
Display totalUnits with appropriate heading
Display totalSales with appropriate heading
```

```
void showTotals(const double sales[], const int units[], int num)
{
    int totalUnits = 0;
    double totalSales = 0.0;
    for (int index = 0; index < num; index++)
    {
        totalUnits += units[index];
        totalSales += sales[index];
    }
    cout << "Total Units Sold: " << totalUnits << endl;
    cout << "Total Sales:      $" << totalSales << endl;
}
```

Copyright by Dau Hai Phong - 2023

HỎI ĐÁP

Bài tập

- **Bài tập chương 7:**
  - Chapter7_Programming Challenges.pdf

ĐẠI NAM
UNIVERSITY

Trân trọng cảm ơn!

ĐẬU HẢI PHONG
Giảng viên
dauhaiphong@dainam.edu.vn
0912441435