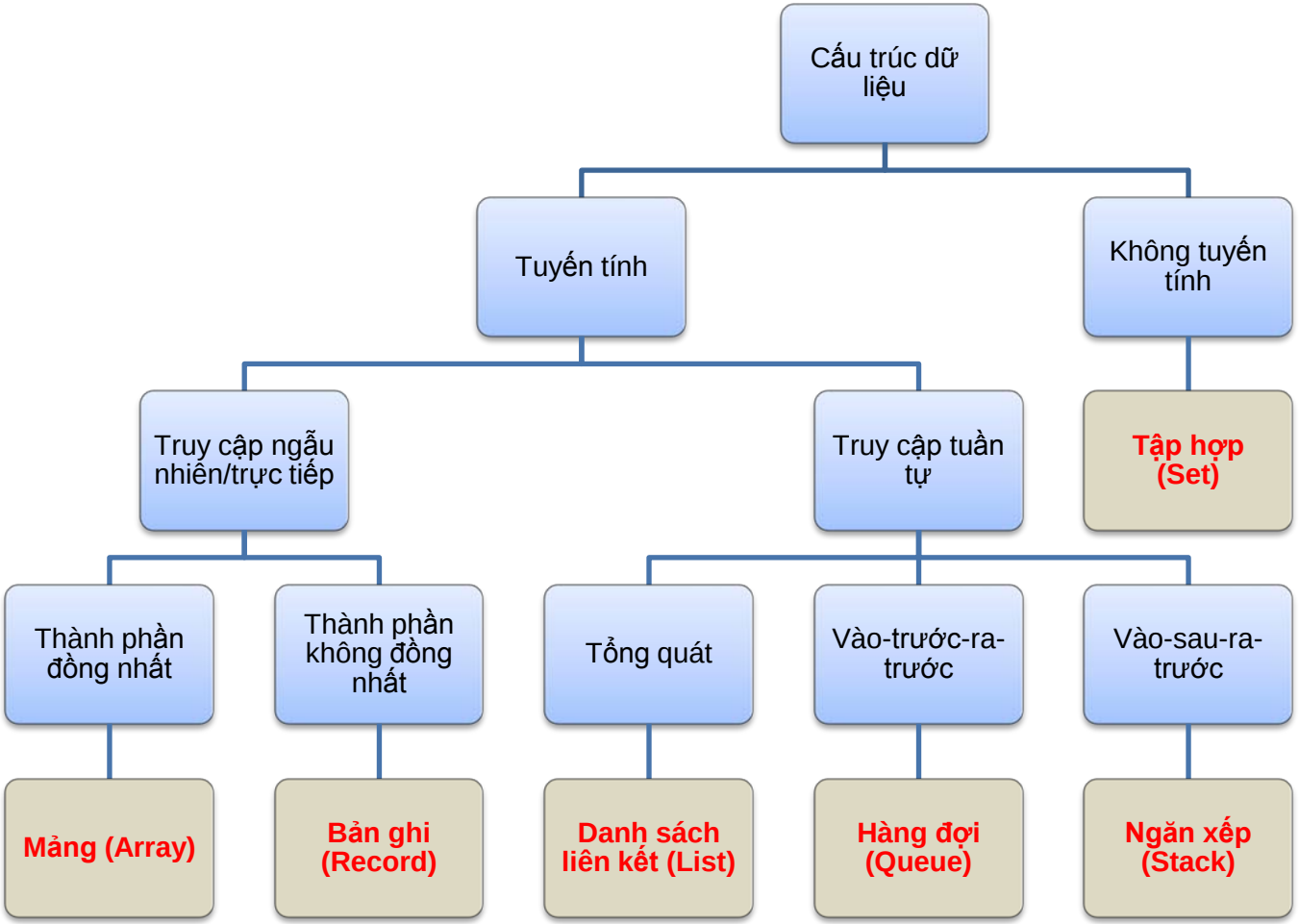
HK I, 2012-2013

Bài 6: Ngăn xếp

Giảng viên: Hoàng Thị Điệp
Khoa Công nghệ Thông tin – Đại học Công Nghệ

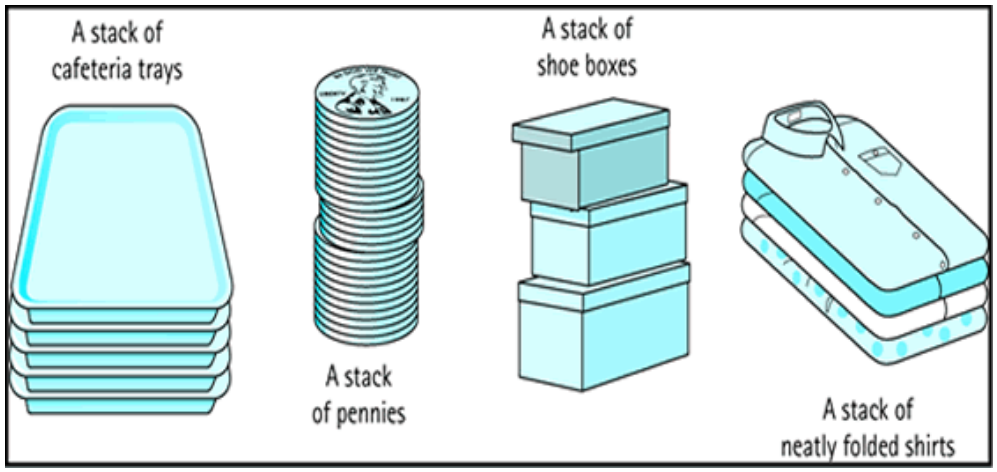
Nguồn tham khảo chính:
http://www.cs.nyu.edu/~melamed/courses/102/lectures/
http://users.encs.concordia.ca/~dssouli/COEN352.html

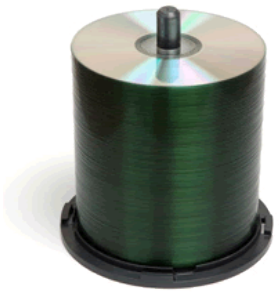
Tổng quan


Cấu trúc dữ liệu
Tuyến tính
Không tuyến tính
Truy cập ngẫu nhiên/trực tiếp
Truy cập tuần tự
Tập hợp (Set)
Thành phần đồng nhất
Thành phần không đồng nhất
Tổng quát
Vào-trước-ra-trước
Vào-sau-ra-trước
Mảng (Array)
Bản ghi (Record)
Danh sách liên kết (List)
Hàng đợi (Queue)
Ngăn xếp (Stack)

Ngăn xếp

- Ngăn xếp là gì?
  - Là một danh sách nhưng các phép toán chỉ được thực hiện ở một đỉnh của danh sách.
- Tính chất
  - Vào trước ra sau (First In Last Out: FILO)





KDLTT ngăn xếp

- Trừu tượng hóa cấu trúc ngăn xếp
  - Đặc tả dữ liệu

    $A = (a_0, a_1, \ldots, a_n)$

    trong đó a_n là đỉnh ngăn xếp
  - Đặc tả các phép toán
    1. Thêm phần tử x vào đỉnh ngăn xếp: push(x)
    2. Loại phần tử ở đỉnh ngăn xếp: pop()
    3. Kiểm tra ngăn xếp có rỗng hay không: isEmpty()
    4. Kiểm tra ngăn xếp có đầy hay không: isFull()
    5. Đếm số phần tử của ngăn xếp: size()
    6. Trả về phần tử ở đỉnh ngăn xếp: top()

Giao diện C++ của KDLTT ngăn xếp

```
template <typename Object>
class Stack {
public:
    int size();
    bool isEmpty();
    Object& top()
          throw(EmptyStackException);
    void push(Object o);
    Object pop()
          throw(EmptyStackException);
};
```

Minh họa các thao tác

thao tác	output	ngăn xếp
push(3)		(3)
push(5)		(3, 5)
pop()		(3)
top()	3	(3)
push(8)		(3, 8)
pop()		(3)
size()	1	(3)
pop()		()
pop()	lỗi: ngăn xếp rỗng	()
push(15)		(15)
top()	15	(15)

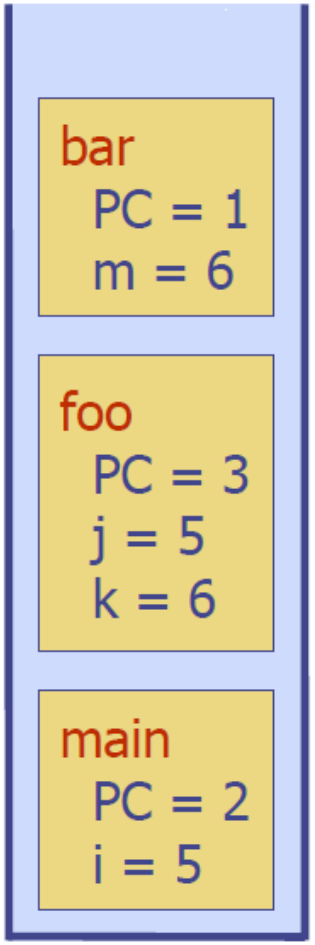
Ứng dụng

- Trực tiếp
  - Nhật trình lướt web lưu trong trình duyệt
  - Chuỗi undo trong một trình soạn thảo văn bản
  - Việc lưu trữ các biến cục bộ khi một hàm gọi hàm khác và hàm này lại gọi tới hàm khác nữa, …
- Gián tiếp
  - Cấu trúc dữ liệu phụ trợ cho các thuật toán
  - Một phần của CTDL khác

Ngăn xếp chạy chương trình của C++

- Hệ thống chạy chương trình của C++ dùng một ngăn xếp để quản lý một chuỗi các hàm đang thực thi
- Khi một hàm được gọi, hệ này push vào ngăn xếp một frame chứa:
  - các biến cục bộ và giá trị trả về
  - con đếm chương trình (program counter) để theo dõi câu lệnh đang được thực hiện
- Khi một hàm trả về gì đó, frame của nó bị pop khỏi ngăn xếp và quyền điều khiển được chuyển cho hàm ở đỉnh ngăn xếp.

```
main() {
  int i;

  i = 5;
  foo(i);
}

foo(int j)
{
  int k;
  k = j+1;
  bar(k);
}

bar(int m)
{
  …
}
```



Cài đặt ngăn xếp bởi mảng

- Có thể cài đặt KDLTT ngăn xếp bằng một mảng một chiều
- Thêm các phần tử từ trái sang phải
- Có một biến để theo dõi chỉ số của phần tử đỉnh ngăn xếp

```
Algorithm size()
    return t + 1

Algorithm pop()
    if isEmpty() then
        throw EmptyStackException
    else
        t ← t – 1
        return S[t + 1]
```

Cài đặt ngăn xếp bởi mảng (2)

- Mảng có thể đầy
- Thao tác push do đó có thể ném ngoại lệ FullStackException
  - Đây là hạn chế của cài đặt bằng mảng
  - Không chỉ xảy ra với ngăn xếp

```
Algorithm push(o)
  if t = S.length – 1 then
    throw FullStackException
  else
    t ← t + 1
    S[t] ← o
```

Cài đặt ngăn xếp bởi mảng C++

```
template <typename Object>
class ArrayStack {
private:
    int capacity;       // stack capacity
    Object *S;          // stack array
    int top;            // top of stack
public:
    ArrayStack(int c) {
        capacity = c;
        S = new Object[capacity];
        t =  –1;
    }
```

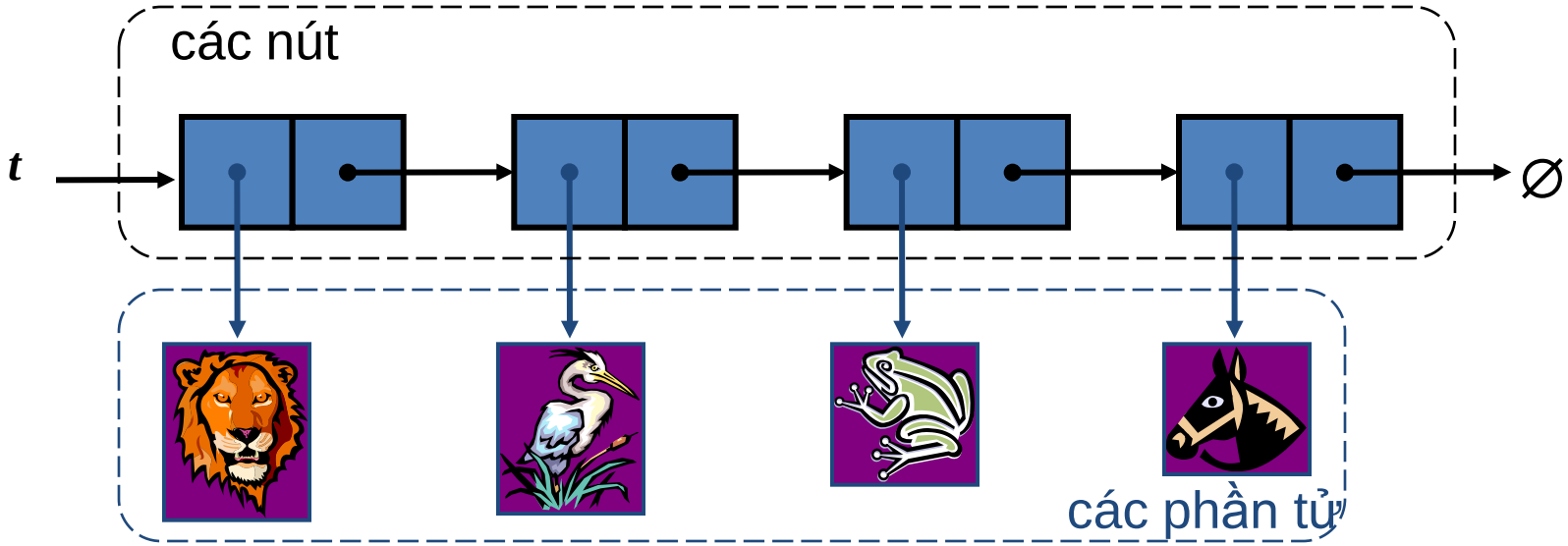
```
    bool isEmpty()
      {  return (t < 0);  }

    Object pop()
          throw(EmptyStackException) {
       if(isEmpty())
          throw EmptyStackException
             ("Access to empty stack");
          return S[t--];
    }
// … (other functions omitted)
```

Hiệu năng và hạn chế

- Hiệu năng
  - Gọi n là số phần tử của ngăn xếp
  - Không gian sử dụng là $O(n)$
  - Mỗi thao tác thực hiện trong thời gian $O(1)$
- Hạn chế
  - Kích thước tối đa của ngăn xếp phải được chỉ định trước và không thể thay đổi
  - Cố push phần tử mới vào ngăn xếp đã đầy sẽ sinh ngoại lệ do cài đặt (implementation-specific exception)

Cài đặt ngăn xếp bởi DSLK

- Có thể cài đặt ngăn xếp bởi một DSLK đơn
- Phần tử đỉnh ngăn xếp được lưu ở nút đầu danh sách
- Không gian sử dụng là $O(n)$ và mỗi thao tác thực hiên trong thời gian $O(1)$



Kiểm tra biểu thức dấu ngoặc cân xứng

- Mỗi ngoặc mở “(“, “[“, “{“ phải được cặp với một ngoặc đóng “)“, “]“, “}“ tương ứng.
- Ví dụ
  - cân xứng: ( )(( )){([( )])}
  - không cân xứng: ((( )(( )){([( )])}
  - không cân xứng:)(( )){([( )])}
  - không cân xứng: ({[ ])}
  - không cân xứng: (

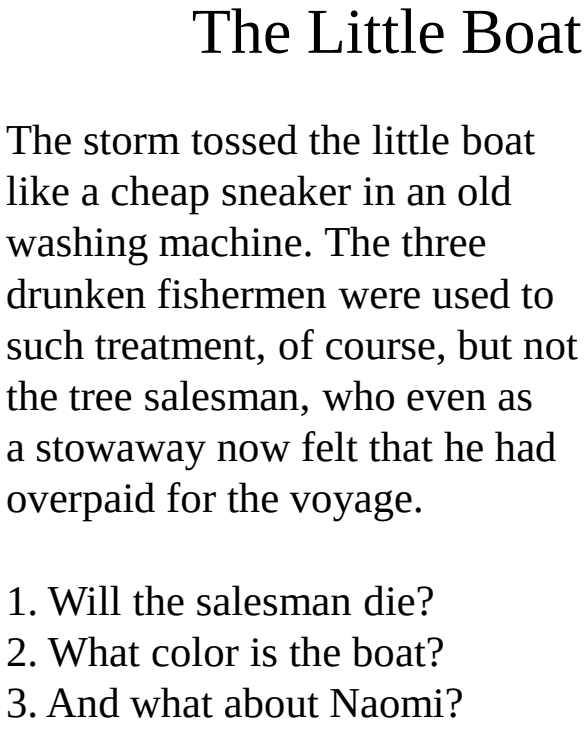
Thuật toán

Algorithm ParenMatch(*X*,*n*):
Input: An array *X* of *n* tokens, each of which is either a grouping symbol, a variable, an arithmetic operator, or a number
Output: **true** if and only if all the grouping symbols in *X* match
Let *S* be an empty stack
for *i*=0 to *n*-1 **do**
    if *X*[*i*] is an opening grouping symbol **then**
        S.push(*X*[*i*])
    else if *X*[*i*] is a closing grouping symbol **then**
        if *S*.isEmpty() **then**
            return false *{nothing to match with}*
        if *S*.pop() does not match the type of *X*[*i*] **then**
            return false *{wrong type}*
if *S*.isEmpty() **then**
    return true *{every symbol matched}*
else
    return false *{some symbols were never matched}*

Kiểm tra thẻ HTML cân xứng

- Mỗi thẻ mở <name> phải được cặp với một thẻ đóng </name> tương ứng

```
<body>
<center>
<h1> The Little Boat </h1>
</center>
<p> The storm tossed the little
boat like a cheap sneaker in an
old washing machine. The three
drunken fishermen were used to
such treatment, of course, but
not the tree salesman, who even as
a stowaway now felt that he
had overpaid for the voyage. </p>
<ol>
<li> Will the salesman die? </li>
<li> What color is the boat? </li>
<li> And what about Naomi? </li>
</ol>
</body>
```



Bài tập

1. Viết chương trình cài đặt cấu trúc ngăn xếp bằng mảng.
2. Viết chương trình cài đặt cấu trúc ngăn xếp bằng danh sách liên kết.
3. Với mỗi phép toán trong câu 1, 2 tính độ phức tạp.
4. Viết chương trình kiểm tra tính hợp lệ các cặp ngoặc ( ) [ ] { } cho một chương trình C++.