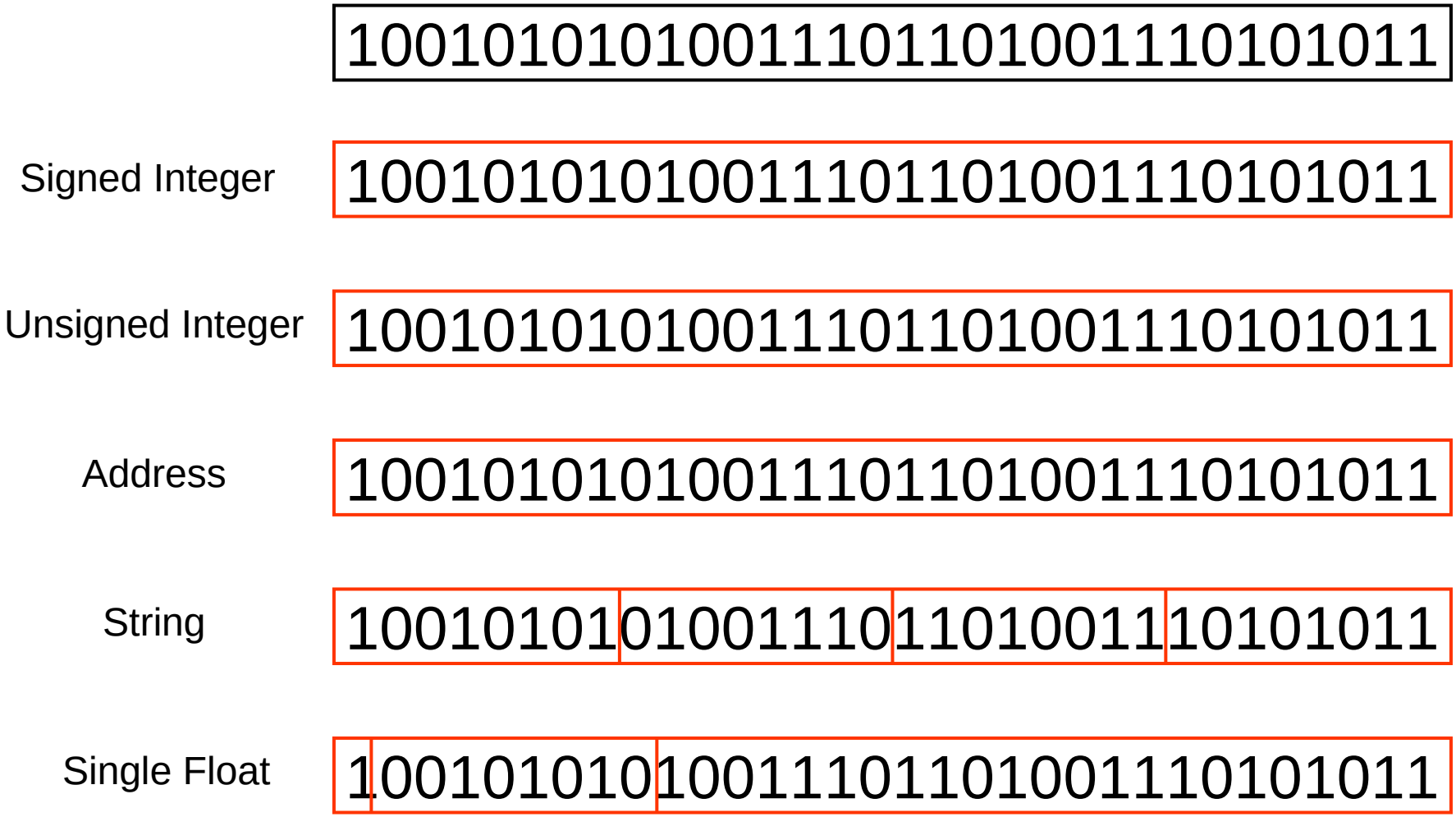
Chương 2 : Kiểu dữ liệu

Bits và giá trị (values)



Nội dung chính của chương

- Giới thiệu kiểu dữ liệu
- Kiểu dữ liệu cơ bản
- Kiểu do người dùng định nghĩa
- Kiểu dữ liệu có cấu trúc

Dữ liệu (data)

- Dữ liệu / máy tính
  - Được lưu trữ ở bộ nhớ
  - Được tổ chức thành dạng bits, bytes, words
- Dữ liệu / NNLT
  - Được tổ chức phức tạp: số, chuỗi ký tự, các mảng, ngăn xếp, …
  - Thuật ngữ **đối tượng dữ liệu** (data objects) là một tập hợp (set) của một hoặc nhiều mẫu dữ liệu
  - VD Int = {- 2^{31} … 0 … 2^{31} }

Đối tượng dữ liệu

- Một đối tượng dữ liệu là một chỗ chứa các giá trị của dữ liệu – một vị trí trong bộ nhớ máy tính, được đặt tên và có thể lưu trữ giá trị của dữ liệu
- Có 1 tập các thuộc tính xác định số lượng và kiểu của các giá trị mà đối tượng dữ liệu có thể lấy, tổ chức luận lý của các giá trị này

Phân loại đối tượng dữ liệu

- Phân loại dựa trên cấu trúc: 2 loại
  - ĐTDL sơ cấp: chứa một giá trị dữ liệu đơn
  - ĐTDL có cấu trúc: tập hợp của các dữ liệu khác
  - VD!
- Phân loại trên nguồn gốc: 2 loại
  - ĐTDL tường minh: do người dùng khai báo như biến, hằng…)
  - ĐTDL ẩn: được định nghĩa bởi hệ thống như các ngăn xếp lưu trữ các giá trị trung gian, các ô nhớ đệm

Thuộc tính của ĐTDL

- Thuộc tính của một ĐTDL là một tính chất đặc trưng của ĐTDL đó
- Mỗi ĐTDL có một tập hợp các thuộc tính để phân biệt ĐTDL này với ĐTDL khác.
- Các ĐTDL sơ cấp có một thuộc tính duy nhất là kiểu dữ liệu
  - VD Int: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
- Các ĐTDL có cấu trúc có thêm các thuộc tính nhằm xác định số lượng, kiểu dữ liệu của các phần tử. VD Struct, Union

Giá trị dữ liệu (GTDL)

- GTDL của một ĐTDL sơ cấp có thể là một số, một ký tự hoặc là một giá trị logic tùy thuộc vào kiểu của ĐTDL đó.
- GTDL được biểu diễn bởi 1 dãy các bits, ĐTDL được biểu diễn bởi một khối ô nhớ.
  - VD một ĐTDL A chứa GTDL B ~ khối ô nhớ biểu diễn A chứa dãy bits biểu diễn B
- GTDL của một ĐTDL có cấu trúc là một tập hợp các GTDL của các phần tử của ĐTDL đó.

Thời gian tồn tại của ĐTDL

- Thời gian tồn tại (lifetime) của một ĐTDL là khoảng thời gian ĐTDL chiếm giữ bộ nhớ của máy tính
- Thời gian này được tính từ khi ĐTDL được tạo ra cho đến khi nó bị hủy bỏ

Biến và hằng

- Biến
  - ĐTDL được định nghĩa và đặt tên một cách tường minh
  - Biến là cách trù tượng hóa một hoặc nhiều ô nhớ
  - Thuộc tính : name, address, value, type, lifetime, scope
- Hằng
  - ĐTDL có tên và giá trị của hằng không thay đổi trong thời gian tồn tại của nó
  - VD!

Phạm vi của biến (Scope)

- Phạm vi của biến là vùng mà biến có thể trông thấy được
- Phạm vi tĩnh (static scope or block) : C, C++, Java & C#

```
void sub(){
    int count;
    …
    while (…) {
        int count;
       count++;
    }
}
```

```
int main(){

  int count;

  count = 10;
  printf("count %d\n",count);

  {
     int count;
     count = 2;
     count++;
     printf("count %d\n",count);
  }
  count = count -5;
  printf("count %d\n",count);

}
```

Phạm vi của biến (tt)

- Phạm vi động (dynamic scope): Perl, COMMON, LISP cho phép khai báo phạm vi động

```
Procedure Big is
    X : integer;
    Procedure Sub1 is
        Begin
        … X …
        End;
    Procedure Sub2 is
        X: integer;
        Begin
        …
        End
    Begin
    …
    End
```

Phạm vi của biến: ví dụ

```
x : integer;
procedure setX(n : integer)
    x := n;
procedure printX()
    write integer(x);
procedure first()
    setX(4); printX();
procedure second()
   x : integer; setX(2); printX();
// bắt đầu chương trình chính
setX(1);
second();
printX();
first();
printX();
```

```
int x = 2
procedure f(){    x = x * x;}
procedure g(){     h(f) }
procedure h(P){
    int x = 4
    P()
}
// bắt đầu chương trình chính
g()
print(x)
```

Scope VS Lifetime của biến

- Đôi khi scope và lifetime là tương tự nhau
  - Java: 1 biến được khai báo trong 1 hàm (no method calls), scope được bắt đầu từ vị trí khai báo đến cuối hàm và lifetime là thời gian thực hiện của hàm
- Nhưng scope và lifetime hoàn toàn không tương tự nhau
  - Scope là khái niện liên quan đến không gian
  - Lifetime là khái niện liên quan đến thời gian

Các liên kết (binding)

- Một ĐTDL có thể tham gia vào nhiều mối liên kết trong thời gian tồn tại của nó
- Các mối liên kết phổ biến
  - ĐTDL với GTDL: phép gán
  - ĐTDL với 1 hoặc nhiều tên tham chiếu: khai báo, gọi và trả CT con
  - ĐTDL với ĐTDL (component): giá trị của pointer và pointer có thể bị thay đổi do thay đổi pointer
  - ĐTDL và ô nhớ: trình quản lý bộ nhớ (routine) thiết lập

Đặt tên (names)

- Cách đặt tên : biến, hằng và hàm
  - Có phân biệt ký tự HOA và thường không?
  - Từ khóa gồm những từ nao?
- Chiều dài tối đa của tên
  - Fortran cho phép tối đa 31
  - C99 chấp nhận 63 ký tự đầu tiên
  - Java không giới hạn độ dài
- Hầu hết các NNLT đều cho phép dùng các chữ cái, các số và «_» để đặt tên

Đặt tên (names) (tt)

- Vài NNLT dùng khái niện «camel» : ký tự HOA giữa 2 từ. VD myStack
- C, C++, Java phân biệt giữa chữ hoa chữ thường (case sensitive). VD rose, ROSE và Rose
- Từ khóa trong từng NNLT được dùng trong các tình huống đặc biệt
- Không được dùng từ khóa để đặt tên, chẳng hạn tên biến, tên hằng, tên hàm, etc

Đặc tả kiểu dữ liệu

- Đặc tả kiểu dữ liệu bao gồm ba yếu tố sau:
  - Các thuộc tính: xác định các đối tượng dữ liệu thuộc kiểu
  - Các giá trị: mà các đối tượng dữ liệu thuộc kiểu có thể được gán cho
  - Các tác vụ: các thao tác có thể có trên các đối tượng dữ liệu thuộc kiểu

Kiểu dữ liệu

- Kiểu dữ liệu là một tập hợp các ĐTDL và tập hợp các phép toán thao tác trên các ĐTDL đó.
  - VD -int là valid và -string là invalid
- Các kiểu dữ liệu nguyên thuỷ (primitive types)
  - Số : Int, Real (float point), decimal
  - Char (character)
  - Boolean (Pascal)

Kiểu dữ liệu (tt)

- Kiểu dữ liệu có cấu trúc (or composite types)
  - Records (Struct)
  - Variant Records (Unions)
  - Arrays
  - Strings
  - Pointers
  - Lists (Perl)
  - Files
  - Kiểu do người dùng đinh nghĩa, VD boolean trong C

Khai báo kiểu

- Khai báo là một phát biểu trong chương trình dùng để chuyển tới bộ dịch, thông tin về số lượng, kiểu của ĐTDL và tên đối tượng dữ liệu; cần thiết trong quá trình thực hiện chương trình
- Sự khai báo có thể chỉ rõ thời gian tồn tại của ĐTDL
- Một số NNLT không cần khai báo biến trước khi sử dụng
  - Ưu điểm : mềm dẽo, khuyết điểm : khó quản lý

Ưu điểm của kiểu dữ liệu

- Cung cấp ngữ cảnh hoàn hảo (**implicit context**) cho các tác vụ, do đó programmer không cần chỉ ra ngữ cảnh cụ thể cho các tác vụ
- Cho phép trình biên dịch phát hiện toàn bộ các lỗi thông thường của programmer
  - Kiểm tra kiểu không thể ngăn ngừa hết các tác vụ vô nghĩa

Kiểm tra kiểu

- Kiểm tra kiểu là kiểm tra xem mỗi tác vụ được thực hiện có nhận được đối số thích hợp thuộc kiểu dữ liệu thích hợp không.
- Cùng kiểu (Type Equivalence): khi hai giá trị có cùng kiểu hay không?
- Sự tương thích kiểu (Type Compatibility): khi giá trị của kiểu A được dùng trong phạm của kiểu B hay không?
- Sự quy nạp kiểu (Type Inference): kiểu thức có kiểu gì, hoặc phép toán trả kiểu gì?

Kiểm tra kiểu (tt)

- Các luật kiểm tra kiểu
  - Kiểu của các biến : int, double, hoặc boolean
  - Biểu thức toán :
    - Tham biến (arguments) : int hoặc double
    - Kết quả : integer nếu các đối số là integer và double, ngoặc ngược lại
  - Biểu thức quan hệ
    - Tham biến (arguments) : integer hoặc double
    - Kết quả : lý luận (boolean)
  - Phép gán : biến và biểu thức đước gán phải có cùng kiểu
  - If và While, etc : kiểu của điều kiện phải là boolean

Kiểm tra kiểu: động và tĩnh

- Kiểm tra kiểu động : kiểm tra kiểu được thực hiện **trong khi thực hiện chương trình**
  - Sử dụng đuôi kiểu (type tag) được lưu trong từng ĐTDL
  - VD: ĐTDL kiểu số nguyên sẽ chứa cả giá trị số nguyên và đuôi kiểu ‘integer’
  - Các ngôn ngữ SNOLBOL4, LISP và APL đòi hỏi kiểm tra kiểu động
  - Ưu điểm: linh hoạt (không cần khai báo kiểu, kiểu của các biến có thể thay đổi trong khi chạy nếu cần thiết)
  - Khuyết điểm: có thể sót lỗi về kiểu, khó debug, tốn nhiều bộ nhớ, tốc độ chương trình chậm lại

Kiểm tra kiểu: tĩnh

- Kiểm tra kiểu tĩnh: chương trình dịch sẽ kiểm tra kiểu lúc **dịch**
  - Đòi hỏi các thông tin:
    - Đối với mỗi tác vụ: số lượng, thứ tự và kiểu dữ liệu của các đối số và kết quả
    - Đối với mỗi tên biến: kiểu của ĐTDL có tên biến đó
    - Đối với mỗi hằng: kiểu của ĐTDL có tên hằng đó
  - Ưu điểm: kiểm tra tất cả các tác vụ, không cần đuôi kiểu, chương trình chạy nhanh hơn
  - Khuyết điểm: không mền dẻo

Kiểm tra kiểu: mạnh và yếu

- Kiểm tra kiểu mạnh: NNLT không cho phép các tác vụ mà dữ liệu không tương thích nhau
- Kiểm tra kiểu yếu : Cho phép thực hiện chuyển kiểu dữ liệu một cách tự động
  - VD
    - NN C Int A; double B; B = 10.1; A = B; valid
    - NN C++ Int A; double B; B = 10.1; A = B; Warning

Thông tin về kiểm tra kiểu vài NN

	Kiểm tra mạnh	Kiểm tra tĩnh
C++/C#/Java	√	√
C	**x**	√
Perl	**x**	**x**
Python	√	**x**
Pascal/Modula/Ada	√	√
Common Lisp	√	**x**

Chuyển kiểu (convertion)

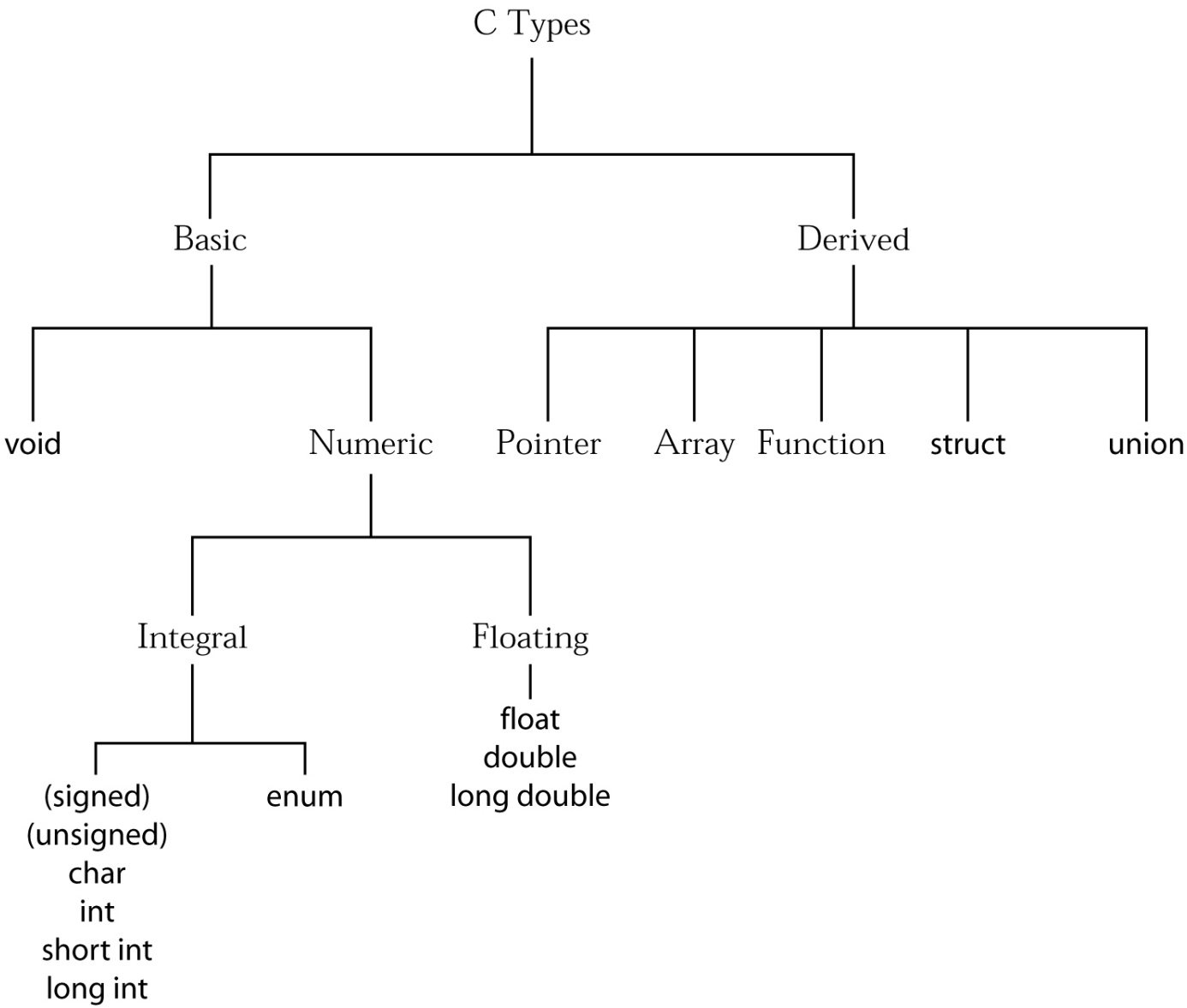
- Nếu không có sự thương thích kiểu lúc kiểm tra kiểu, NNLT thực hiện một trong hai tác vụ sau
  - Sự không tương thích kiểu sẽ bị báo lỗi hoặc
  - Một sự chuyển đổi kiểu tự động được thi hành để chuyển đổi kiểu
- Hầu hết NNLT đều cung cấp 2 cách chuyển kiểu
  - Tập hợp các hàm để chuyển kiểu. VD, Pascal có hàm Round
  - Sự chuyển kiểu ngầm (thực hiện ngầm). VD int = int + double;

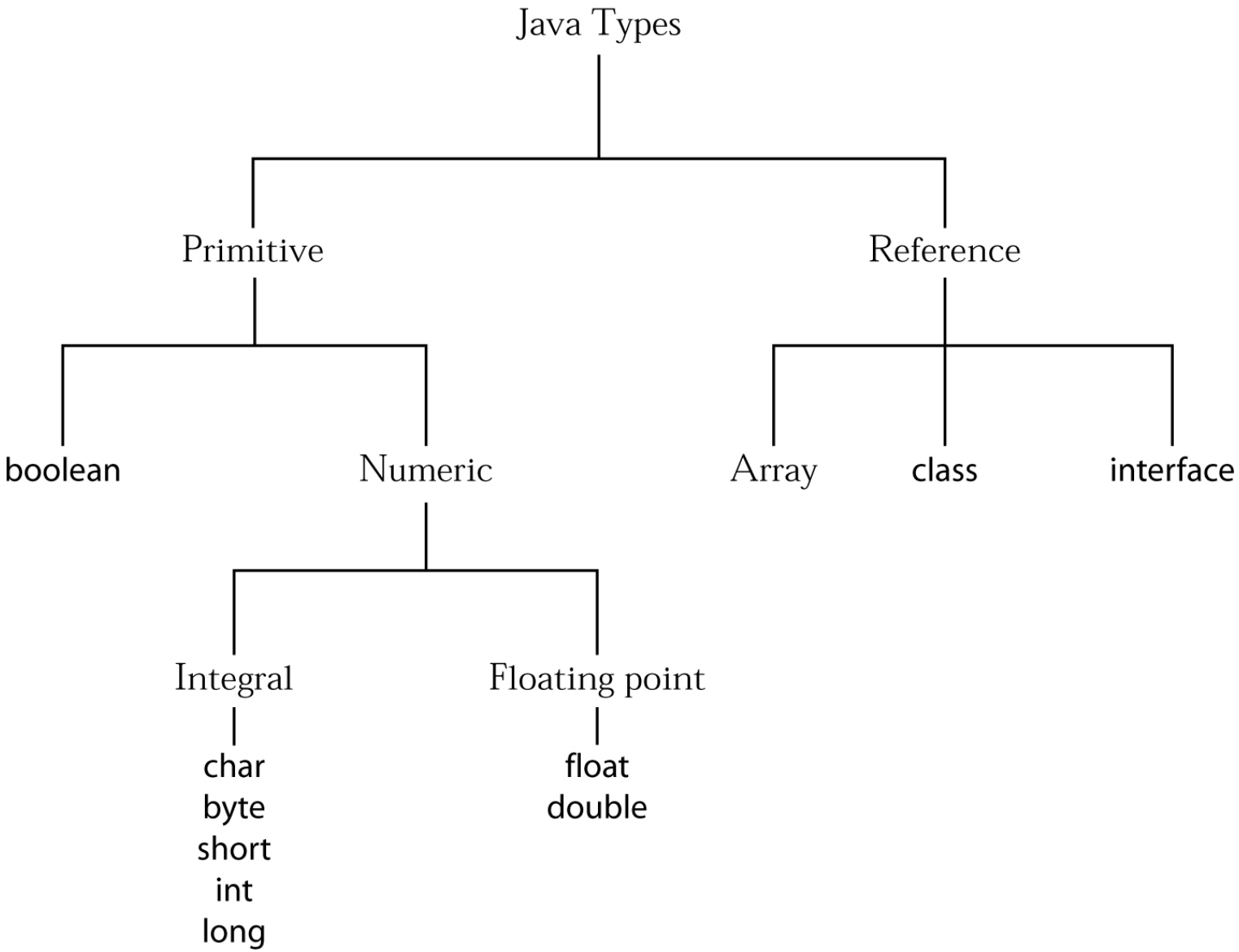
Chuyển kiểu (tt)

- Chuyển kiểu chỉ thực hiện được giữa các kiểu gần nhau :
  - Int to float và ngược lại
  - Enum to Int và ngược lại
  - subclass to base class
- Nonconverting Cast: giữa 2 kiểu có cùng kích thước lưu trữ
  - Pointer to integer
  - Integer or enum to pointer
  - Giữa 2 pointer trỏ tới 2 đối tượng khác kiểu
  - Giữa 2 pointer trỏ tới 2 hàm khác kiểu

Sự tương hợp kiểu

- Hai kiểu T_1 và T_2 tương hợp nếu ĐTDL kiểu T_1 có thể xuất hiện ở vị trí của ĐTDL kiểu T_2 và ngược lại:
  - Có thể gán ĐTDL T_1 cho ĐTDL T_2 và ngược lại
- Hai qui tắc xác định tương hợp
  - Tương đương tên
  - Tương đơn cấu trúc
    - Type T1=array[1..20] of integer
      T2=array[1..20] of integer


C Types
Basic
Derived
void
Numeric
Pointer
Array
Function
struct
union
Integral
Floating
(signed)
(unsigned)
char
int
short int
long int
enum
float
double
long double


Java Types
Primitive
Reference
boolean
Numeric
Array
class
interface
Integral
Floating point
char
byte
short
int
long
float
double

Kiểu dữ liệu cơ bản

- Kiểu số
  - Số nguyên (integer, int)
  - Số thực dấu chấm động (floating-point real)
- Kiểu logic (Boolean)
- Kiểu ký tự (char or character)

Kiểu số nguyên

- Số nguyên dài/ngắn, VD Java : byte, short, int, long; với C : short, int, long
- Số nguyên có dấu (signed) hoặc không dấu; bit bên trái miêu tả dấu 0: positive; 1 negative

  1001010101001110110100111010101 1
- Các phép toán
  - BinOp(Binary operations) : +, -, x, /, mod
  - UnaryOp : âm (-), dương (+)
  - RelOp (relational Operations) : equal, not equal, less-than, greater-than, less-than-or-equal
  - BitOp (Bit Operations) : and (&), or (|), shif bits (<<)

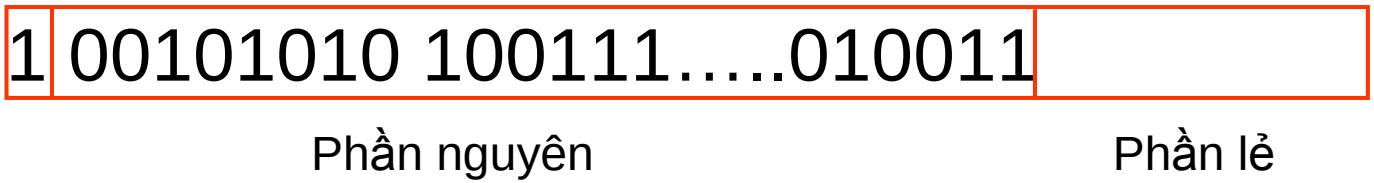
Số thực dấu chấm động

- Dãy số có thứ tự từ một số âm nhỏ nhất tới một số lớn nhất được xác định trước, nhưng dấu phẩy thập phân không cố định
  - VD 1.234567, 123456.7, 0.00001234567
- Hai thành phần biểu diễn giá trị của số thực trong ô nhớ : phần định trị (mantissa) và phần mũ (exponent)
  - Single : 1 bit dấu, 23 bits phần trị, 8 bits mũ
  - double : 1 bit dấu, 53 bits phần trị, 10 bits mũ

Single Float

Số thực dấu chấm cố định

- Được biểu diễn bằng một chuỗi các chữ số có chiều dài cố định
- Có dấu chấm thập phân phân cách phần nguyên và phần lẻ
- VD khai báo trong COBOL: X picture 999 v 99
- Được lưu trữ ở dạng nhị phân biểu diễn trị số của phần nguyên và phần lẻ



Kiểu lý luận (logic)

- Kiểu logic chỉ nhận 2 giá trị : đúng (TRUE) và sai (FALSE)
- Trong Pascal, boolean được xem là kiểu liệt kê, Boolean = (FALSE, TRUE) và FALSE<TRUE
  - Trong Pascal : **type** Boolean = (FALSE, TRUE)
  - Trong C : typedef enum {FALSE=0, TRUE} Boolean;
- Các phép toán logic : AND, OR, NOT
- Kiểu logic chỉ dùng duy nhất 1 bit để biểu diễn giá trị 0 = FALSE, 1 = TRUE

Kiểu ký tự

- Kiểu ký tự được lưu trong máy tính ở dạng mã số
- Mã số này được lưu trữ trong 8-bits (ASCII)
- Bảng mã Unicode (16 bit) được đưa vào Java, C#
- Các phép toán
  - Các phép toán quan hệ (relational operations)
  - Phép gán

Kiểu do người dùng định nghĩa

- Kiểu liệt kê (Set)
- Kiểu miền con số nguyên (subrange)

Kiểu liệt kê

- NNLT Pascal và Ada cho phép người dùng đặt ra kiểu dữ liệu bằng cách liệt kê các giá trị của kiểu đó
  - VD type NGAY = (CN, Hai, Ba, Tu, Nam, Sau, Bay);
- Giá trị của ĐTDL kiểu liệt kê được biểu diễn bằng các số nguyên : 0,1,2,…
  - VD kiểu NGAY cần sử dụng 7 giá trị từ 0 đến 6 :
- Lợi ích của kiểu liệt kê: tăng khả năng dễ đọc và tính dễ viết và độ tin cậy của ngôn ngữ.

Miền con của số nguyên

- Kiểu dữ liệu mà tập các giá trị là một dãy các giá trị nguyên trong một khoảng giới hạn đã định.
  - VD A : 1..10 (Pascal); A : Integer Rang 1..10 (Ada)
- Lợi ích của kiểu miền con
  - Tiết kiệm ô nhớ về mặt lưu trữ. VD miền con 1..10 chỉ yêu cầu 4 bits các giá trị trong miền con
  - Dể kiểm tra kiểu hơn số nguyên. VD Month: 1..12, thi lệnh gán Month=0 là không hợp lệ

Kiểu dữ liệu có cấu trúc

- Mãng
- Mẫu tin
- String
- Pointer
- Tập tin

Mãng : một chiều (vector)

- Mãng một chiều: ĐTDL bao gồm một số **cố định** các phần tử có kiểu **giống nhau** được tổ chức thành một dãy tuần tự.
- Các thuộc tính của mãng 1 chiều
  - Số lượng các phần tử
  - Kiểu dữ liệu của mỗi một phần tử
  - Chỉ số của mỗi phần tử, VD trong Pascal

    var myArray: array[-10..10] of integer; chỉ số -10..10

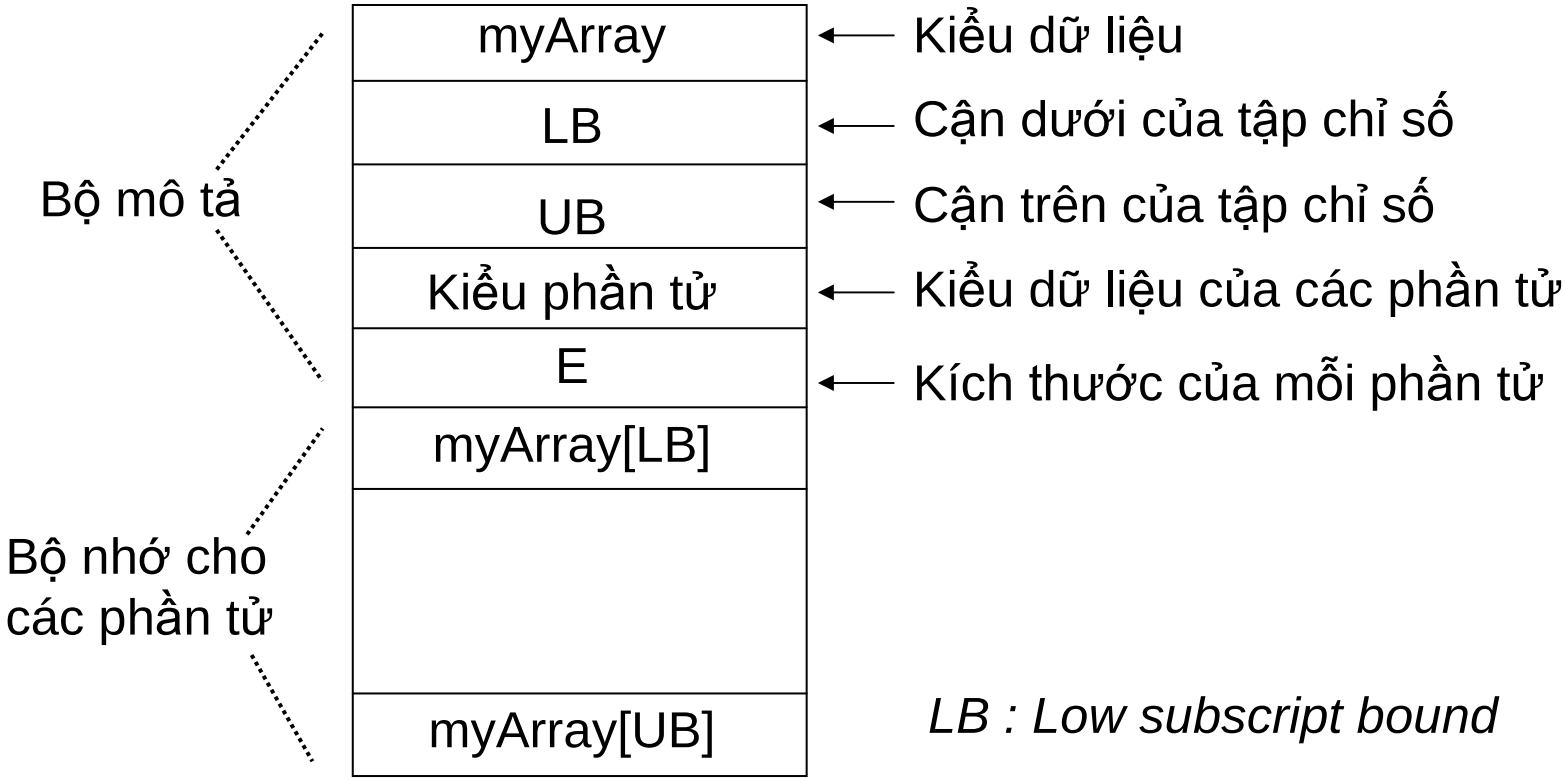
    trong C: int myArray[20]; chỉ số 0..19

Mãng : một chiều (tt)

- Các phép toán
  - Lựa chọn 1 phần tử [], VD myArray[4]
  - Các phép toán khác : gán 2 mãng, các phép toán số học trên từng cập 2 mãng có cùng kích thước
- Sự khởi tạo mãng, trong C

  Int list [] = {4,5,7,83}; char name[] = «NoName»;
- Hầu hết các NNLT cấp phát các phần tử của mãng liên tục trong bô nhớ

Mãng : một chiều (tt)



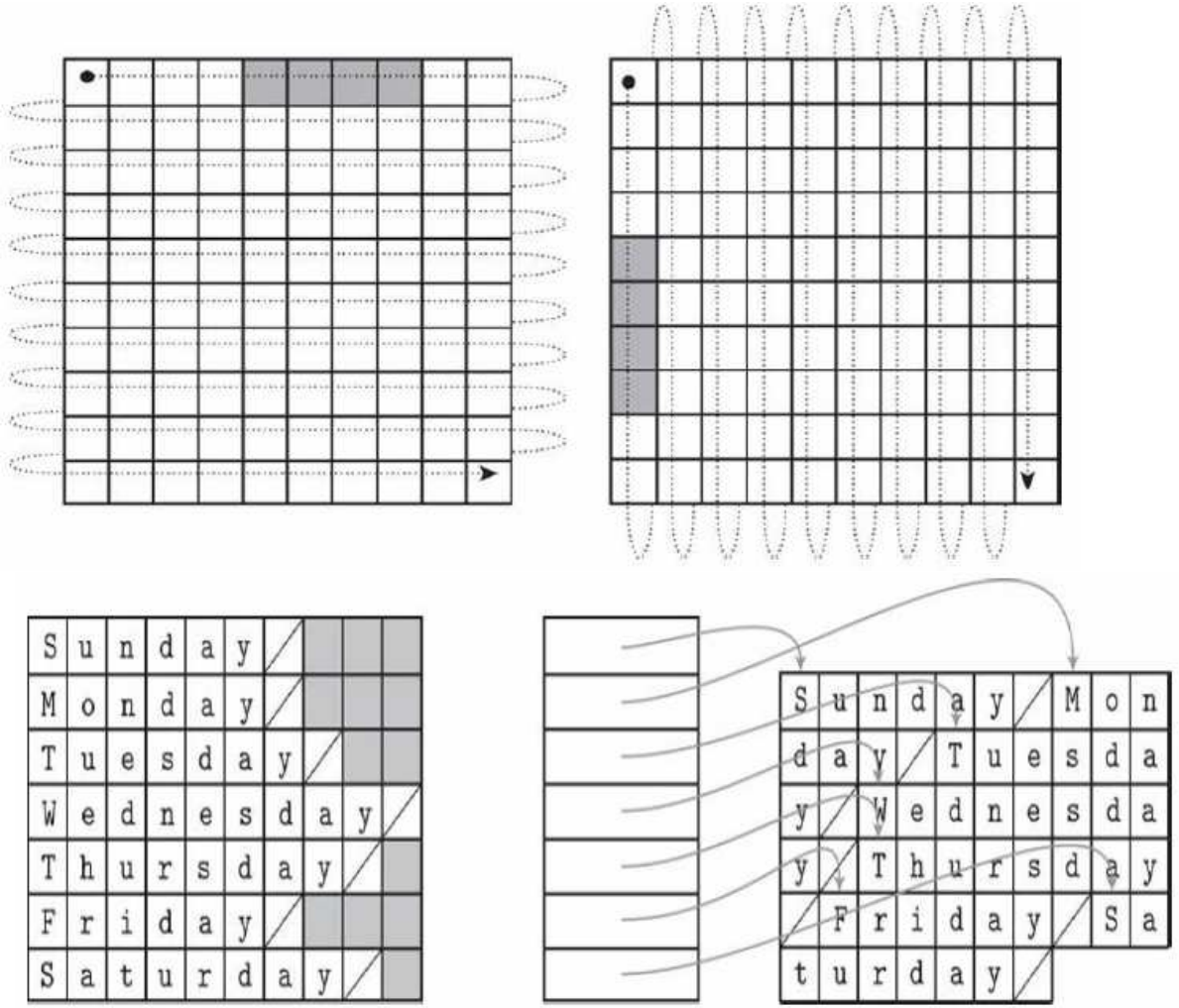
LB : Low subscript bound

UB: Upper subscript bound

Mãng 2 hoặc nhiều chiều

- Mãng 2 chiều là mãng của mãng 1 chiều; mãng 3 chiều là mãng của mãng 2 chiều.
- Mãng n chiều có n tập các phần tử
  - Pascal myArray array[a..b,c..d] of real;
  - C float myArray[b-a][d-c];
- Sự biểu diễn bộ nhớ
  - Lưu trữ theo trật tự dòng (C, C++, C#)
  - Lưu trữ theo trật tự cột (Fortran)
  - Lưu trữ theo pointer của dòng (Java)

Mãng 2 hoặc nhiều chiều (tt)



Mẫu tin

- ĐTDL cho phép chứa 1 số nhất định các phần tử có kiểu khác nhau
- Có nhiều thuật ngữ khác nhau
  - Algol 68, C, C++, and Common Lisp: **struct**
  - Java, C++, C#: **class**
  - Pascal: **record**
- Các thuộc tính
  - Số phần tử, kiểu của các phần tử
  - Tên phần tử

Mẫu tin (tt)

- VD

In Pascal:

```
type two_chars =
  packed array [1..2] of char;

type element=record
  name : two_chars;
  atomic_number : integer;
  atomic_weight : real;
  metallic : Boolean
end;
```

In C:

```
struct element {
    char name[2];
    int atomic_number;
    double atomic_weight;
    _Bool metallic;
};
```

Mẫu tin có cấu trúc thay đổi

```
type PayType = (Salaried, hourly)
Var Employee: record
    ID : integer;
    Dept : array [1..3] of char;
    Age : integer
    case Payclass: Paytype of
        Salaried : (MonthRate : real;
                    StartDate : integer);
        Hourly: (HourRate : real;
                 Reg : integer;
                 Overtime : integer;
    end
```

Mẫu tin có cấu trúc thay đổi (tt)

<table>
<tr><td colspan="2">ID</td></tr>
<tr><td colspan="2">Dept</td></tr>
<tr><td colspan="2">Age</td></tr>
<tr><td colspan="2">PayClass</td></tr>
<tr><td>MonthlyRate</td><td>HourRate</td></tr>
<tr><td>StartDate</td><td>Reg</td></tr>
<tr><td></td><td>Overtime</td></tr>
</table>

PayClass = Salaried PayClass = Hourly

Kiểu chuỗi ký tự

- Chuỗi ký tự là mãng của các ký tự, nhưng có thêm các tác vụ khác mà các mãng khác không có
- C, Pascal, Ada yêu cầu chiều dài của chuỗi giá trị không vượt quá một giới hạn đã được khai báo
- Lisp, Java, C# cho phép thay đổi chiều dài của biến có kiểu chuỗi ký tự
- SNOBOL4 không giới hạn chiều dài của biến kiểu chuỗi

Kiểu chuỗi ký tự (tt)

- Các phép toán
  - Phép nối kết, Pascal sử dụng toán tử «+»
  - Các phép toán quan hệ thông thường : bằng, lớn hơn, nhỏ hơn
  - Chọn chuỗi con bằng chỉ số vị trí, VD Fortran str(6:10)
  - Chọn chuỗi con bằng so mẫu (substr của C)
  - Định dạng xuất / nhập

Con trỏ (pointer)

- Cấp phát bộ nhớ (cấp phát) là dành riêng các ô nhớ cho CT sử dụng
- Cấp phát tĩnh là sự cấp phát ô nhớ cho ĐTDL được thực hiện trong qua trình dịch
  - Ưu điểm : dể sử dụng
  - Nhược điểm : không tối ưu
- Cấp phát động là cấp phát lúc thực hiện CT
  - Ưu điểm : sử dụng bộ nhớ 1 cách tối ưu
  - Nhược điểm : người dùng phải tự quản lý

Con trỏ (tt)

- Con trỏ là ĐTDL chứa địa chỉ khối ô nhớ được cấp phát động
- Ô nhớ cấp phát động được tham chiếu bằng con trỏ
- Đặt tả thuộc tính
  - Con trỏ chỉ có thể tham chiếu đến các ĐTDL cùng kiểu (Pascal, Ada)
  - Con trỏ có thể tham chiếu tới các ĐTDL có kiểu nhau khác

Con trỏ (tt)

- Các phép toán
  - Cấp phát ô nhớ động

    C++: int *x = new int;

    C: int *x = (int *) malloc(sizeof(int));

    Java: BankAccount b = new BankAccount(0);
  - Truy xuất ô nhớ

    *x = *x + 1;
  - Thu hồi ô nhớ

    C++: delete x;

    C: free(x);

Con trỏ - kiểu đệ qui

- struct IntList{
  int data;
  struct IntList next;
  };
- struct IntList{
  int data;
  struct IntList *next;
  };
- Yêu cầu con trỏ cuối trỏ đến NULL để kết thúc

Tập tin

- Tập tin có 2 thuộc tính : lưu trữ trong bộ nhớ ngoài và thời gian tồn tại lâu
- Tập tin có thể được truy nhập theo một trong hai mode: READ hoặc WRITE
- Các phép toán chủ yếu đối với tập tin
  - Mở tập tin
  - Đọc tập tin
  - Ghi tập tin
  - Đóng tập tin

Tập tin : Pascal

- Khai báo biến tập tin

  <biếnTậpTin> = FILE OF <Kiểu phần tử>;

- Mở tập tin :

  RESET(biếnTậpTin)

- Đọc tập tin:

  READ(biếnTậpTin, b1, b2, ..., bN)

- Ghi tập tin :

  WRITE(biếnTậpTin, b1, b2, ..., bN):

- Đóng tập tin

  CLOSE(biếnTậpTin)

Tận tin : C

- Khai báo biến tập tin

  FILE *<FP>

- Mở tập tin

  FP = fopen("filename","mode"); mode : r, w, a, rw

- Đọc tập tin

  fread (ptr,size,n,FP); char *ptr; int n,size;

- Ghi tập tin

  Fwrite (ptr,size,n,FP); char *ptr; int n,size;

- Đóng tập tin

  Fclose (FN)