Bài 5: KDLTT danh sách cài đặt bằng mảng động

Giảng viên: Hoàng Thị Điệp

Khoa Công nghệ Thông tin – Đại học Công Nghệ

Nội dung chính

1. Thư viện khuôn mẫu chuẩn STL
2. Con trỏ và bộ nhớ động C++
3. KDLTT danh sách cài bằng mảng động
   - Bộ ba quan trọng
   - Cải tiến hàm insert, append
4. Ứng dụng KDLTT danh sách
   - Tập động
   - Đa thức
   - Ma trận thưa

Mã nguồn minh họa 2 phần đầu được lấy và chỉnh sửa từ cplusplus.com

diepht@vnu

Thư viện khuôn mẫu chuẩn STL

- <array>
- <vector>
- <deque>
- <forward_list>
- <list>
- <stack>
- <queue>
- <priority_queue>
- <set>
- <multiset>
- <map>
- <multimap>
- <unordered_set>
- <unordered_multiset>
- <unordered_map>
- <unordered_multimap>
- <bitset>
- <valarray>

diepht@vnu

Khuôn mẫu (template)

```
// khai báo thư viện...
int getMaxI(int a, int b){
    int result;
    result = (a > b)? a : b;
    return (result);
}
double getMaxD(double a, double b){
    double result;
    result = (a > b)? a : b;
    return (result);
}
int main(){
    int i=5, j=6, k;
    double l=10.3, m=5.1, n;
    k=getMaxI(i,j);
    n=getMaxD(l,m);
    cout << k << endl;
    cout << n << endl;
    return 0;
}
```

```
// khai báo thư viện...
template <class T>
T getMax(T a, T b){
    T result;
    result = (a > b)? a : b;
    return (result);
}

int main(){
    int i=5, j=6, k;
    double l=10.3, m=5.1, n;
    k=getMax<int>(i,j);
    n=getMax<double>(l,m);
    cout << k << endl;
    cout << n << endl;
    return 0;
}
```


diepht@vnu

Ví dụ thư viện <vector>:: push_back()

```
// vector::push_back
#include <iostream>
#include <vector>
#include <conio.h>
using namespace std;

int main()
{
    vector<int> myvector;
    int myint;

    cout << "Nhap vao cac so nguyen (nhap 0 de dung):\n";
    do{
        cin >> myint;
        myvector.push_back(myint);
    }while(myint);

    cout << "myvector chua " << int(myvector.size()) << " so nguyen.\n";
    getch();
    return 0;
}
```


diepht@vnu

Ví dụ thư viện <vector>::insert()

```
// vector::insert
// #include cac thu vien can thiet ....
int main(){
    vector<int> myvector(3,100);
    vector<int>::iterator it;
    it = myvector.begin();
    it = myvector.insert(it, 200);
    myvector.insert(it, 1, 300);

    // "it" khong con hop le, lap 1 gia tri moi:
    it = myvector.begin();
    vector<int> anothervector(2, 400);
    myvector.insert(it + 1, anothervector.begin(), anothervector.end());

    int myarray[] = {501, 502, 503};
    myvector.insert(myvector.begin(), myarray, myarray + 3);

    cout << "myvector chua day so:";
    for(it = myvector.begin(); it < myvector.end(); it++)
        cout << ' ' << *it;
    cout << '\n';
    getch();
    return 0;
}
```


diepht@vnu

Ví dụ thư viện <vector>::erase()

```
// vector::erase
// #include cac thu vien can thiet ....
int main(){
    vector<int> myvector;
    // them 1 vai gia tri ban dau (tu 1 den 10)
    for(int i = 1; i <= 10; i++) myvector.push_back(i);

    // xoa phan tu thu 6
    myvector.erase(myvector.begin() + 5);
    // xoa 3 phan tu dau tien:
    myvector.erase(myvector.begin(), myvector.begin() + 3);

    cout << "myvector chua day so:";
    for(unsigned i = 0; i < myvector.size(); ++i)
        cout << ' ' << myvector[i];

    cout << '\n';
    getch();
    return 0;
}
```


diepht@vnu

Nội dung chính

1. Thư viện khuôn mẫu chuẩn STL
2. Con trỏ và bộ nhớ động C++
3. KDLTT danh sách cài bằng mảng động
   - Bộ ba quan trọng
   - Cải tiến hàm insert, append
4. Ứng dụng KDLTT danh sách
   - Tập động
   - Đa thức
   - Ma trận thưa

Con trỏ và bộ nhớ động

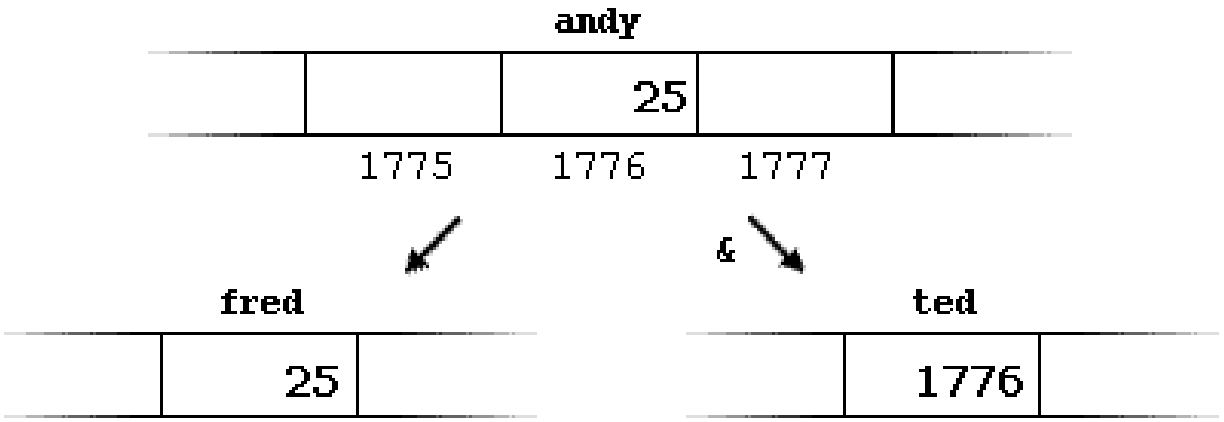
- Các vấn đề chính
  - Bộ nhớ máy tính
  - Biến và địa chỉ của biến
  - Biến con trỏ
  - Mảng và con trỏ
  - Bộ nhớ động: cấp phát và giải phóng
  - Mảng động và con trỏ
  - Truyền tham số là con trỏ
- Tham khảo
  - Chương 6, Giáo trình: Lập trình cơ bản với C++
    - Tác giả: Trần Minh Châu, Lê Sỹ Vinh, Hồ Sĩ Đàm,
  - http://www.cplusplus.com/doc/tutorial/pointers/
  - Day 8, Teach Yourself C++ in 21 Days

Toán tử lấy địa chỉ (&)

andy = 25;

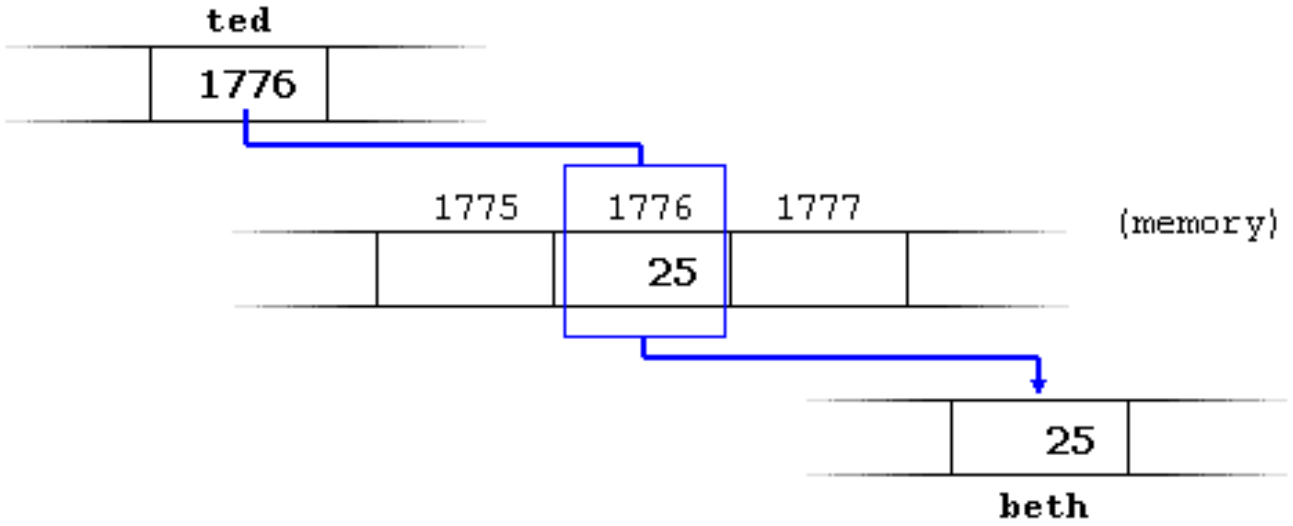
fred = andy;

ted = &andy;



Toán tử giải tham chiếu (*)

beth = *ted;



Khai báo biến con trỏ: VD1

```
#include <iostream>
#include <conio.h>
using namespace std;
int main(){
    int v1, v2;
    int * p;
    p = &v1;
    *p = 10;
    p = &v2;
    *p = 20;
    cout << "v1 = " << v1 << endl;
    cout << "v2 = " << v2 << endl;
    getch();
    return 0;
}
```

Khai báo biến con trỏ: VD2

```
#include <iostream>
#include <conio.h>
using namespace std;
int main(){
    int v1 = 5, v2 = 15;
    int * p1, * p2;
    p1 = &v1; // p1 = dia chi cua v1
    p2 = &v2; // p2 = dia chi cua v2
    *p1 = 10; // gia tri cua bien tro boi p1 = 10
    *p2 = *p1; // gtri cua bien tro boi p2 =
               // gtri cua bien tro boi p1
    p1 = p2; // p1 = p2 (value of pointer is copied)
    *p1 = 20; // gia tri cua bien tro boi p1 = 20
    cout << "v1 = " << v1 << endl;
    cout << "v2 = " << v2 << endl;
    getch();
    return 0;
}
```


diepht@vnu

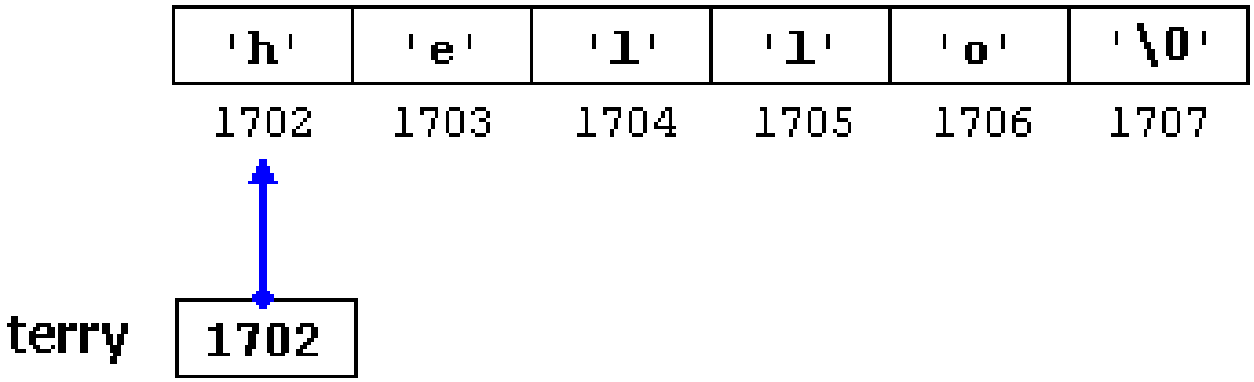
Con trỏ và mảng

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{
    int numbers[5];
    int * p;
    p = numbers;  *p = 2;
    p++;  *p = 3;
    p = &numbers[2];  *p = 5;
    p = numbers + 3;  *p = 7;
    p = numbers;  *(p+4) = 9;
    for(int n = 0; n<5; n++)
        cout << numbers[n] << ", ";
    getch();
    return 0;
}
```


diepht@vnu

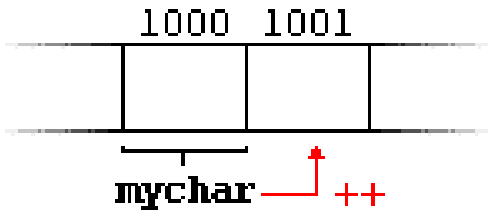
Khởi tạo con trỏ

const char * terry = "hello";

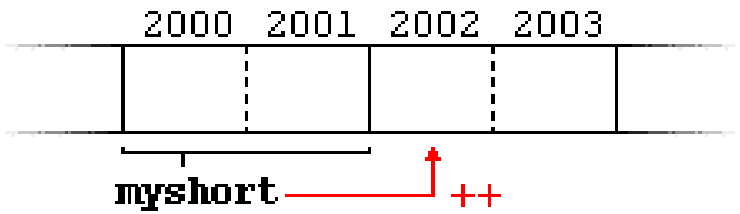


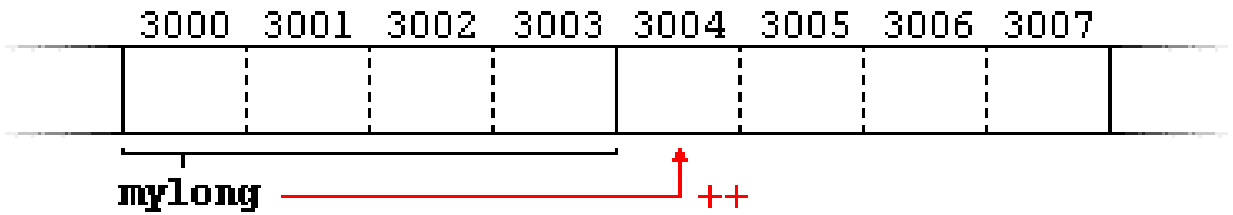
Phép toán số học trên con trỏ

```
char *mychar;
short *myshort;
long *mylong;
```



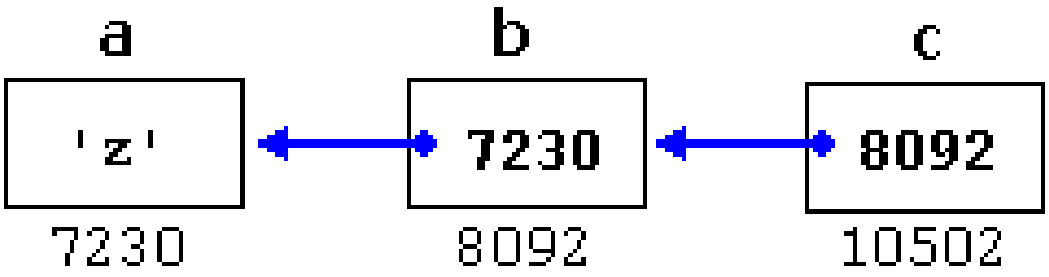

```
mychar++;
myshort++;
mylong++;
```





Con trỏ tới con trỏ

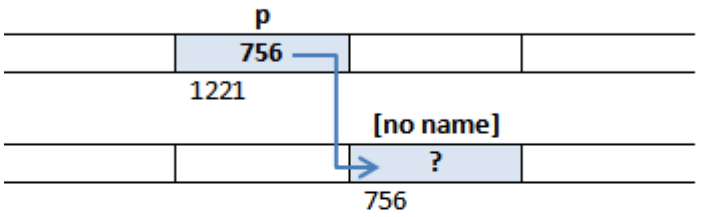
```
char a;
char * b;
char ** c;
a = 'z';
b = &a;
c = &b;
```



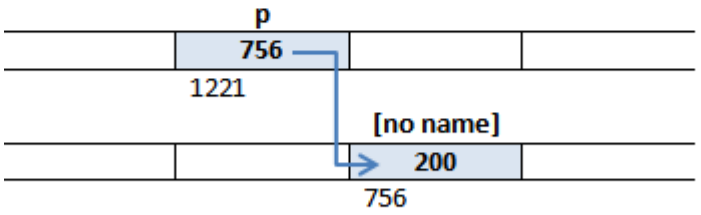
new và delete biến đơn

int * p;

p = new int;



***p = 200;**



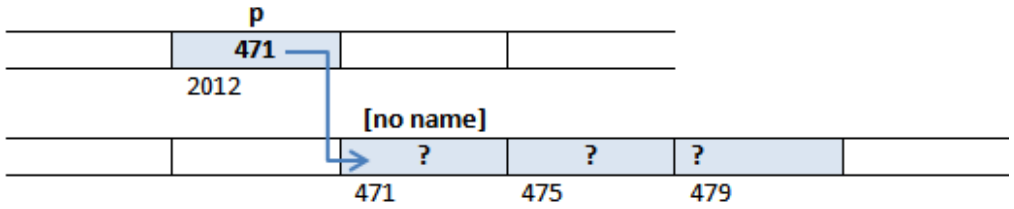
delete p;

diepht@vnu

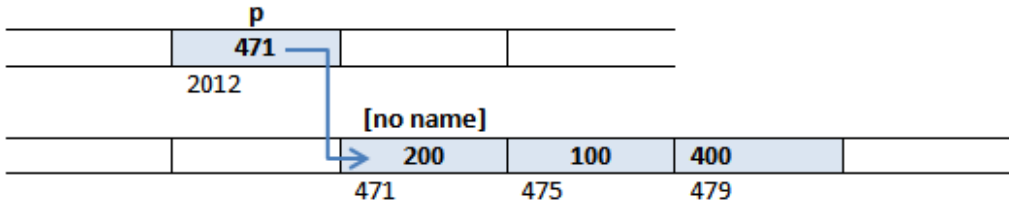
new và delete mảng

```
int * p;
```


```
p = new int[3];
```




```
*p = 200; p[1] = 100; *(p + 2) = 400;
```




```
delete [] p; p = 0;
```

Nội dung chính

1. Thư viện khuôn mẫu chuẩn STL
2. Con trỏ và bộ nhớ động C++
3. KDLTT danh sách cài bằng mảng động
   - Bộ ba quan trọng
   - Cải tiến hàm insert, append
4. Ứng dụng KDLTT danh sách
   - Tập động
   - Đa thức
   - Ma trận thưa

KDLTT danh sách cài bằng mảng C++

Cấp phát tĩnh

```
template <class Item>
class List{
public:
   static const int MAX = 50;
   // ...
private:
   Item element[MAX];
   int last;
};
```

Cấp phát động

```
template <class Item>
class Dlist{
public:
   // ...
private:
   Item * element;
   int size;
   int last;
};
```


diepht@vnu

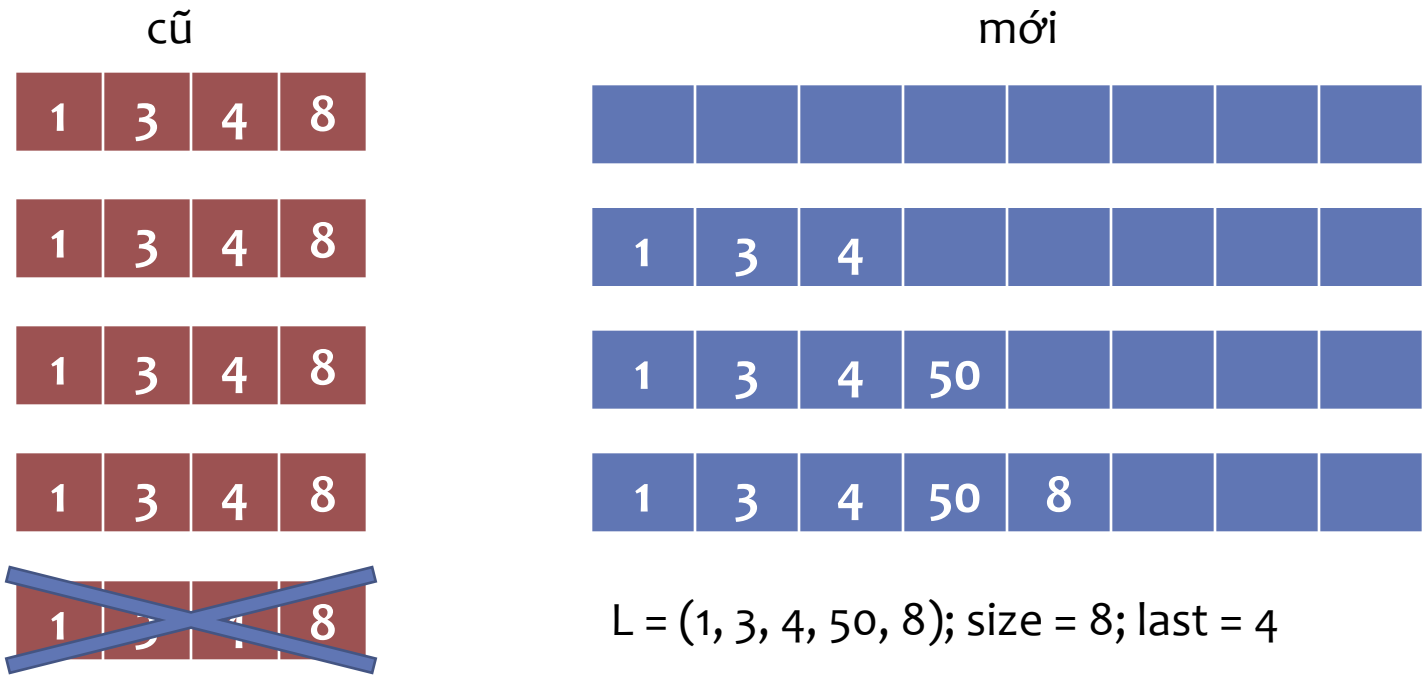
Bộ ba quan trọng

Dlist có thành phần dữ liệu cấp phát động nên phải cài đặt bộ ba

1. Hàm kiến tạo sao chép
   - copy constructor
2. Toán tử gán
   - operator=
   - overloading operator
3. Hàm hủy
   - destructor

Hàm insert, append của Dlist

- L = (1, 3, 4, 8); size = 4; last = 3
- insert(L, 3, 50)



L = (1, 3, 4, 50, 8); size = 8; last = 4

Hàm insert, append của Dlist

Khi mảng đầy

- Cấp phát động một mảng mới có cỡ gấp đôi mảng cũ
- Chép đoạn đầu của mảng cũ sang mảng mới
- Đưa phần tử cần xen vào mảng mới
- Chép đoạn còn lại của mảng cũ sang mảng mới
- Hủy mảng cũ
- Cập nhật size, last

Viết mã C++!

Nội dung chính

1. Thư viện khuôn mẫu chuẩn STL
2. Con trỏ và bộ nhớ động C++
3. KDLTT danh sách cài bằng mảng động
   - Bộ ba quan trọng
   - Cải tiến hàm insert, append
4. Ứng dụng KDLTT danh sách
   - Tập động
   - Đa thức
   - Ma trận thưa

Ứng dụng KDLTT danh sách

- **Tập động**
  - Mỗi phần tử có thành phần khóa phân biệt. Các giá trị khóa có quan hệ thứ tự
  - Các phép toán: empty, insert, erase, seach, getMax, getMin
  - Ví dụ: (("An", 1985), ("Bình", 1986), ("Cường, 1985), ("Dung", 1987))
  - Cài bằng danh sách được sắp hay không được sắp thì tốt hơn?

Bài tập

- Dùng ngôn ngữ C++, viết khai báo lớp IntSet cài đặt KDLTT tập động các số nguyên.

Ứng dụng KDLTT danh sách

- **Đa thức**
  - Ví dụ: ((17,5), (-25, 2), (14, 1), (-32, 0)) biểu diễn đa thức $17x^5 - 25x^2 + 14x - 32$
  - Các phép toán: cộng, trừ, nhân

Ứng dụng KDLTT danh sách

- **Ma trận thưa**
  - Ma trận chỉ chứa một số ít các phần tử khác 0
  - Cách biểu diễn:
    - Xem ma trận như danh sách các dòng
    - Mỗi dòng là một danh sách biểu diễn các phần tử khác 0
    - Mỗi phần tử khác 0 là một cặp (chỉ số cột, giá trị)
  - Ví dụ: (((2, 7), (5, 3)), ((3, 8)), (), ((2, 5), (4, 9)))
  - Các phép toán trên ma trận, trên dòng, trên phần tử

diepht@vnu

Tìm kiếm nhị phân

Algorithm *binarySearch(x, A, first, last)*:
Input: search keyword ***x***
and array ***A*** of Items with two ends marked by indexes ***first*** and ***last***
Output: **true** if ***x*** in ***A***, **false** otherwise

```
if first > last then
    return false
mid ← (first + last) / 2
if x = A[mid].key then
    return true
else if x < A[mid].key then
    binarySearch(x, A, first, mid - 1)
else
    binarySearch(x, A, mid + 1, last)
```


diepht@vnu

Tìm kiếm nhị phân

A →	1	3	4	6	8	9	11
chỉ số →	0	1	2	3	4	5	6

A = (1, 3, 4, 6, 8, 9, 11); x = 4.

search(x, A).

- binarySearch(x, A, first, last)
- binarySearch(x, A, 0, 6)
  - So x với A[3]. x nhỏ hơn.
- binarySearch(x, A, 0, 2)
  - So x với A[1]. x lớn hơn.
- binarySearch(x, A, 2, 2)
  - So x với A[2]. Bằng. Trả về true.

diepht@vnu