bool foo(Data \*p, char\*name) // 이항 함수

{

if(strcmp(p->GetName(), name))

return false;

else

return true;

}

// 이항함수 ==> 단항 함수로 변경 : bind의 개념(bind는 함수 객체만 사용가능)

// 함수를 함수 객체로 변경 : function adapter(함수 아답타)

void\* Manager::DataSearch(const char\*name, bool b /\*=true\*/)

{

vector<Data\*>::iterator p1;

p1 = find\_if(vec.begin(), vec.end(), bind2nd( ptr\_fun(foo), name));

if( p1 == vec.end())

return NULL;

if( b )

return \*p1;

else

return p1;

}

void Manager::Search()

{

char name[20];

fflush(stdin);

cout << "검색 이름 : "; cin.getline(name,20);

Data \*p = (Data\*)DataSearch(name);

if( p)

p->DataPrint();

else

cout << "데이터 없다." << endl;

}

======================================

#include <iostream>

using namespace std;

#include <vector>

#include <algorithm>

class Data

{

char name[20];

int age;

public:

Data() {}

Data( char\*\_name, int \_age)

{

strcpy(name, \_name);

age = \_age;

}

char\* GetName() { return name; }

};

vector<Data\*> vec;

bool foo(Data \*p, char\*name) // 이I항¡¿ 함O수ùo

{

if(strcmp(p->GetName(), name))

return false;

else

return true;

}

bool DataSearch1( char\*name, vector<Data\*>::iterator & p1)

{

p1 = find\_if(vec.begin(), vec.end(), bind2nd( ptr\_fun(foo), name));

if( p1 == vec.end())

return false;

return true;

}

void main()

{

vec.push\_back(new Data("김¾e길¾©¡동ì¢¯", 20));

cout << vec.size() << endl;

vector<Data\*>::iterator p;

bool bb = DataSearch1("김¾eㅍ´¨ö길¾©¡동ì¢¯", p);

if( !bb )

{

cout << "못¬©ª찾¡Ì음¨ö" << endl;

}

else

{

vector<Data\*>::iterator it = vec.erase(p);

delete \*it;

cout << vec.size() << endl;

}

// vec.erase(p);

//vector<Data\*>::iterator p1 = vec.erase(p);

}