자료구조 중간고사 (2013.10.17)

1. 추상화와 information hiding의 관계를 설명하고 왜 필요한지를 설명하라.

추상화는 복잡한 시스템을 사용자의 관점에서 필요한 상세만으로 표현

Information hiding은 모듈이나 구조의 상세에 접근을 제어할 목적으로 기능이나 자료구조의 상세를 감추는 것. IH은 ADT를 실제적으로 구현하는 방법 중의 하나. IH은 시스템이나 프로그램 모듈을 복잡한 구현방법에 대한 지식 없이도 쉽게 사용할 수 있도록 복잡한 구조를 사용자로부터 감추는 역할 수행. 추상화는 IH를 통해서 도출된 high level 상세를 이용하여 시스템이나 모듈을 표현함으로써 시스템의 쉽게 사용할 수 있도록 함.

1. 독립적인 iterator class가 필요한 이유를 설명하라.

독립적인 iterator object들을 생성하여 하나의 리스트를 여러 곳에서 동시에 스캔할 수 있게 함.

1. Test pattern은 프로그램 verification과정의 성패에 크게 영향을 미친다. Test pattern을 생성이 중요한 이유를 설명하라.

복잡한 프로그램은 각기 다른 구조를 가진 요소들로 구성된다. 각기 다른 구성요소를 모두 확인하기 위해서는 각 구성요소를 활성화 시킬 수 있는 입력자료가 필요

1. 함수에 Queue를 Pass by value로 전달했을 때 발생하는 문제점과 해결방안을 설명하라.

함수 내에서 dequeue할 경우 dangling pointer 발생. Pass by reference를 이용하거나 deep copy가 되도록 copy constructor를 재정의.

1. Big-O 표기법으로 다음 함수의 복잡도를 표현하라. **계산과정이 없으면 0점 처리(**10)

Fun(int data[N], int N)

{ int sum=0, i, j,k;

For (i=0; i<N; i++)

{ For (j=i; j<N; j++) sum += data[i]\*data[j];

For (j=0; j<10000; j++) {

for (k=0; k<N-200; k++) sum += data[k]\*data[k]; } } }

F(n)= n(n-1)/2 + 10000(n-200)= 0.5n2 -0.5n+ 10000n2 -2000000n = O(n2).

1. 배열을 이용한 sorted list의 Retrieve함수를 binary search를 이용하여 구현하라.

Class SortedType { **public**: int LengthIs(); void RetrieveItem(ItemType& item, bool found); **private:** int length; ItemType info[MAX\_ITEMS]; };

Void SortedType::Retrieve(ItemType& item, bool found)

{int start,end,mid;

Start = 0; end= LengthIs()-1;

Found=False;

While(!Found ) {

mid =(start+end)/2;

If (info[mid]==item) { item=info[mid]; Found=True;}

Else {

if (info[mid]<item) start=mid+1;

else end = mid-1;

if (start>end) return;

}}}

1. 실습시간과 수업시간에 배운 singly linked list를 Header와 trailer를 이용해서 구현하고자 한다. 자료는 오름차순으로 정렬되고 ItemType의 멤버함수로 < 연산자가 정의되어 있다. Header와 trailer는 리스트의 맨 앞과 맨 뒤에 배정된 dummy 노드로써 위치를 고정시키는 역할을 한다.

Class SortedType {

**public**:

SortedType();

~SortedType();

int LengthIs();

void Insert(ItemType item, bool found);

**private:**

int length;

NodeType<ItemType> \*header, \*trailer;

};

* 1. Constructor 함수를 SortedType()을 구현하라. [힌트] header와 trailer는 Header node와 trailer 노드의 포인터이다.

SortedType::SortedType()

{ header = new NodeType<ItemType>;

Trailer = new NodeType<ItemType>;

Header->next = trailer; trailer->next = NULL;

Length=0;}

* 1. Insert()함수를 구현하라. [힌트] header와 trailer가 가리키는 노드는 고정된다. Singly linked list이기 때문에 현재노드와 이전노드를 가리키는 포인터가 필요.

Void SortedType::Insert(ItemType item, bool found = false){

Nodetyep<itemtype> node;

Node.data = item;

Node.next = NULL;

nodeType<itemtype>\* pre;

nodeType<itemtype>\* cur;

pre = header;

cur = header->next;

if(length == 0){

pre->next = node;

node.next = cur;

length++;

found = true;

}

Bool moretosearch = (cur->next != NULL);

while(!found&&moretosearch){

else if(cur->data <item){

pre->next = node;

node.next = cur;

length++;

found = true

}

Else{

Pre = cur;

Cur = cur->next;

}

}

Void SortedType::Insert(ItemType item){

NodeType<ItemType> \*cur, \*prev;

Prev = header;

Cur = header->next;

While(cur !=trailer && cur->info<item){

prev=cur;

cur = cur->next;}

temp= new NodeType<ItemType>;

Temp->info = item;

Prev->next= temp;

temp->next=trailer; // or temp->next= cur;

}