자료구조 중간고사 (2013.04.18)

1. 다음 물음에 답하라.(50)
   1. 다음 두 측면에서 Object-Oriented Programming의 장점을 설명하라.
      1. 공동작업(여러 사람이 협업으로 큰 응용 프로그램을 작성할 때)(7)
         1. Operation과 domain의 정의만으로 독립적인 작업이 가능
         2. 다른 part의 구현과 무관하게 프로그램 작성이 가능
      2. 재사용(8)
         1. Operation과 domain을 하나로 묶어서 독립적인 모듈 생성
         2. 구현방법이 바뀌어도 응용프로그램은 영향을 받지 않음
   2. 자동변수가 저장되는 메모리 공간을 설명하고 이와 관련하여 자동변수의 유효범위를 설명하라. (10점)
      1. Activation Record에 allocation되면 runtime stack에 저장된다.
      2. 해당 변수가 선언된 블록을 벗어나면 소멸된다.
   3. 함수호출과 관련된 dangling pointer 발생상황과 대처방안을 설명하라.(10점)
      1. Pointer 변수를 pass by value로 pass할 때 함수내에서 포인터가 가리키는 메모리를 해지하고 리턴하면 발생
      2. Pass by reference로 전달하거나 deep copy를 해서 전달
   4. Pass by value의 문제점과 해결방안을 설명하라.(10점)
      1. 함수에서 결과를 리턴할 수 없음
      2. Pass by reference로 전달
   5. 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<N-100; j++) {

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

N(N-1)/2 + N(N-100)(N-200)= N3-(300-0.5) N2+… 🡪 O(N3)

1. Header와 trailer를 이용하고, doubly linked structure를 사용하는 Stack class에 대한 다음 물음에 답하라.
   1. Constructor를 구현하라.(header와 trailer node 고려)(5 점)

DLList:: DLList()

{ header = new Node<ItemTyp>; trailer=new Node<ItemType>;

Header->pre = trailer->next=Null; header->next=trailer;

trailer->pre=header; }

* 1. Destructor를 구현하라. (10 점)

DLList:: ~DLList(){

Node<itemType>\* tmp=header;

While (header !=NULL) {

Header = header->next; delete tmp; tmp= header;}

* 1. Push 함수를 구현하라. (10점)

DLList:: Push(ItemType item) {

Node<itemType> \*tmp=new Node<itemType>;

If (!IsFull()) {

Tmp->info=item; tmp->prev = header; tmp->next=header->next;

Header->next->prev = tmp; header->next=tmp;}

}

1. 음악파일 관리 프로그램에 대한 다음 물음에 답하라.
   1. Application class의 ADT를 제시하라. (10)
   2. Item에 입력된 곡명으로 음악파일을 검색하고 기존 레코드를 Item을 통해 입력된 레코드로 교체하는 application class의 멤버함수 ReplaceItemByName(ItemType& item)을 구현하라. 단 음악파일은 다음과 같은 sorted list에 저장되어 있다.

Class **SortedLisType** {

public:

int **IsFull**();

void **ResetList**();

int **GetCurrentItem**(ItemType& item);

void **Replace**(ItemType item);

}; (10점)

Int Application:: ReplaceItemByName(ItemType& item){

itemtype temp;

list.ResetList();

while(list.GetCurrentItem(temp)){

if(temp == item){

list.Replace(item);

Return 1;

}

}

Return 0;

}

Int Application::ReplaceItemByName(ItemType& item)

{ ItemType curItem;

mList.ResetList();

while( mList.GetCurrentItem(curItem))

{ if (curItem==item) { mList.Replace(item); Return 1} }

Return 0}