INHA UNIVERSITY

ICE2004 자료구조론

9

제 목

자료구조론 과제 3 보고서

보고서 작성 서약서

1. 나는 타학생의 보고서를 베끼거나 여러 보고서의 내용을 짜집기하지 않겠습니다.
2. 나는 보고서의 주요 내용을 인터넷사이트 등을 통해 얻지 않겠습니다.
3. 나는 보고서의 내용을 조작하지 않겠습니다.
4. 나는 보고서 작성에 참고한 문헌의 출처를 밝히겠습니다.
5. 나는 나의 보고서를 제출 전에 타학생에게 보여주지 않겠습니다.

나는 보고서 작성시 윤리에 어긋난 행동을 하지 않고 정보통신공학인으로서 나의 명예를 지킬 것을 맹세합니다.

2021년 11월 14일

학부

학년

성명 성시열

학번

[개요]

- 파일은 ArrayVector.h 헤더파일과 DLinkedList.h 헤더파일. 총 두 개의 헤더파일과 이 헤더파일의 기능을 활용할 수 있는 main.cpp 드라이버 파일로 구성되어있다.
- ArrayVector.h 헤더파일은 과제 3 중 (1)번 문제인 ArrayVector와 관련된 문제를 위한 헤더파일이다. ArrayVector에서 사용하는 size(), empty(), erase(), insert(), reserve() 등의 멤버함수와 []연산자 오버로딩 함수도 클래스의 멤버함수로 포함하였다.
- DLinkedList.h 헤더파일은 과제 3 중 (2)번 문제인 Doubly Linked List와 관련된 문제를 위한 헤더파일이다. Doubly Linked List에서 사용하는 empty(), front(), back(), addFront(), addBack(), removeFront(), removeBack()을 사용하였고, 이러한 함수들을 실행하는데 필요한 add()와 remove()도 사용하였다.
- 드라이버 파일인 main.cpp 파일은 과제 3 의 (1)번과 (2)번 문제를 구분하고 간결하기 하기 위해 정상 동작하는 것을 보이기 위한 절차를 각각의 함수로 묶어서 코드를 구성하였다. 이를 위해 main함수 시작 전 함수의 prototype을 적어주고, 이후에 해당 함수의 코드를 정의하였다.

[구현상 특징]

- ArrayVector 클래스 내부에서 다른 정수와 element를 구분하기 위해 typedef을 통해 int의 또 다른 이름 (Elem)을 지정해주었다.
- DLinkedList 클래스에서 protected 접근 지정자를 활용해 멤버 함수 전체에서 사용할 수 있는 add()와 remove()를 사용하였다. 또한, DLinkedList의 단위가 노드 단위이기 때문에 DNode 클래스를 생성하고, friend class를 통해 DLinkedList 클래스에서도 DNode의 private 멤버까지 자유롭게 사용할 수 있게 한다.
- main.cpp에서는 앞에 생성한 ArrayVector.h, DLinkedList.h 두 헤더파일을 #include를 통해 두 파일의 변수 및 멤버 함수의 기능을 사용할 수 있게 한다.

[코드 분석]

1. ArrayVector.h

코드, 목적 및 상세설명
`typedef int Elem;`
element가 들어가는 곳에 Elem Data type을 사용하여 구분 지어준다.

`Elem& operator[](int i){`
`return A[i];`
`}`

배열의 원하는 인덱스의 값을 return해주는 []연산자 오버로딩, return값이 배열 안의 element이므로 Elem으로 구분. 배열에서 받아온 정수의 인덱스를 return해줌.

`void erase(int i) {`
`for (int j = i + 1; j < n; j++) {`
`A[j - 1] = A[j];`
`}`
`A[n-1] = 0;`
`n--;`
`}`

i번째 인덱스에 값을 제거하려면 그 뒤에 있는 모든 배열을 하나씩 앞으로 땡겨준다. i번째 값부터 element의 마지막까지 반복한다. A[i+1]이 A[i]값을 덮으면서 없어진다. 이를 A의 가장 끝 값이 그 앞의 값을 덮을 때까지 반복한다.

`void insert(int i, const Elem& e) {`
`if (n >= capacity) {`
`reserve(max(1, 2 * capacity));`
`}`
`for (int j = n - 1; j >= i; j--) {`
`A[j + 1] = A[j];`
`}`
`A[i] = e;`
`n++;`
`}`

용량보다 입력된 element가 많거나 같아질 때, 용량을 두배로 늘려주고 그 늘려준 배열에 값을 대체시켜야 한다. max를 사용한 이유는 두배 늘려준 용량이 1보다 작은 수일 경우 1을 선택하여야 하기 때문이다. for문에서 배열에 값을 넣어줄 때 원하는 위치의 값부터 끝의 값까지 오른쪽으로 이동시켜 공간을 만든 뒤, 값을 넣어줘야 한다. for문이 끝난 뒤 새로운 element를 넣고 싶은 곳에 e 값을 대입함.

```
        void reserve(int N) {
                if (capacity >= N) {
                        return;
                }
                Elem* B = new Elem[N]();
                for (int j = 0; j < n; j++) {
                        B[j] = A[j];
                }
                if (A != NULL) {
                        delete[] A;
                }
                capacity = N;
                A = B;                                          }
```

reserve함수는 용량이 늘어난 배열에 값을 대체해주는 함수이다. 1보다 크고 capacity의 곱하기 2 한 값이 N이 기존 capacity보다 작을 때 return을 만나 함수를 탈출한다. 이후 원래 배열 A를 옮겨둘 곳인 배열 B를 capacity *2의 크기로 마련한다. 배열은 0으로 초기화함. capacity = N에서 capacity에 2를 곱해준 값이 A배열의 크기가 됨. A=B에서는 A를 깔끔하게 비운 뒤 배열 B에 옮긴 값을 다시 A에 넣어준다.

.

```
void showvector() {
        for(int i = 0; i < capacity; i++){
                cout << A[i] << "  ";
        }
}
```

showvector함수는 배열의 모든 값을 일렬로 볼 수 있는 함수이다. 인덱스 가장 처음부터 가장 끝까지 값을 출력한다.

2. DLinkedList.h

코드, 목적 및 상세설명

```
typedef int Elem;
```

element가 들어가는 곳에 Elem Data type을 사용하여 구분지어준다.

```
private:
        Elem elem;
        DNode* prev;
        DNode* next;
        friend class DLinkedList;
};
```

모든 노드는 prev, elem, next으로 구성되어있음.

Elem elem은 element를 나타냄.

DNode* prev는 이전 노드를 나타냄.

DNode* next는 다음 노드를 나타냄.

DLinkedList 클래스가 DNode클래스의 모든 멤버를 사용할 수 있음.

```
~DLinkedList() {
        while (!empty())
                removeFront();
        delete header;
        delete trailer;
}...
```

header->next와 trailer가 같지 않을 경우
header->next인 앞부분을 계속 지워줘서 가운데 노드를 비워준다. 이후 header와 trailer를 delete하여 전체 노드를 소멸시킨다.

```
bool empty() const {
        return (header->next == trailer);
}...
```

header->next == trailer는 가운데 노드가 비어있는 경우임. 이 때 true를 return하고 그렇지 않으면 false 반환

```
void add(DNode* v, const Elem& e) {
        DNode* u = new DNode;
        u->elem = e;
        v->prev->next = u;
        u->prev = v->prev;
        u->next = v;
        v->prev = u;
}
```

add함수는 원하는 값을 Doubly Linked List에 추가하는 함수이다. 동작원리는 노드를 추가하고 들어갈 위치의 앞노드와 뒷노드를 연결시키는 것이다. 이를 위해 앞노드와 새로운 노드를 연결하고, 새로운 노드와 뒷노드를 연결하는 코드를 작성한다.

3. main.cpp

코드, 목적 및 상세설명

```
void run_arrayvector() {
    ArrayVector arrvec(3);

    for (int i = 0; i < 10; i++) {
        cout << "arrayvector [" << i << "] =  " << i+1 << " 삽입" << endl;
        arrvec.insert(i, i+1);
        cout << "삽입 후 배열 출력 : " << endl;
        arrvec.showvector();
        cout << endl;
    }
}
```

3칸의 배열을 생성한 뒤 값을 앞에서부터 차례대로 1부터 넣는다. 3보다 커질경우 용량이 2배로 늘어나고 나머지 4부터 자연스럽게 대입되는 것을 볼 수 있다. 배열을 0으

로 초기화하였으므로 값이 들어가지 않은 곳은 0이 된다. 이후 다시 6칸을 넘겼을 때 용량의 2배인 12칸이 된다.

```
void run_doublylinkedlist() {
    DLinkedList DLL;

    cout << endl << "list 앞에 1 2 3 4 5 삽입" << endl;
    for (int i = 5; i >= 1; i--) {
        DLL.addFront(i);
        cout << "앞 : " << DLL.front() << " / 뒤 : " << DLL.back() << endl;
    }

    cout << endl << "list 뒤에 6 7 8 9 10 삽입" << endl;
    for (int i = 6; i <= 10; i++) {
        DLL.addBack(i);
        cout << "앞 : " << DLL.front() << " / 뒤 : " << DLL.back() << endl;
    }

    cout << endl << "list 앞에 3개 제거" << endl;
    for (int i = 0; i < 3; i++) {
        DLL.removeFront();
        cout << "앞 : " << DLL.front() << " / 뒤 : " << DLL.back() << endl;
    }

    cout << endl << "list 뒤에 3개 제거" << endl;
    for (int i = 0; i < 3; i++) {
        DLL.removeBack();
        cout << "앞 : " << DLL.front() << " / 뒤 : " << DLL.back() << endl;
    }
}
```

첫번째 for문

DLL의 앞에서부터 5 4 3 2 1 순으로 삽입을 한다. 그렇게 되면 처음 넣었던 5가 가장 뒤로 가게 되면서 앞에서부터 1 2 3 4 5가 된다.

두번째 for문

DLL의 뒤에서부터 6 7 8 9 10 순으로 삽입을 한다. 그렇게 되면 1 2 3 4 5가 있던 리스트에 그대로 뒤에서부터 6 7 8 9 10이 들어가 1 2 3 4 5 6 7 8 9 10이 차례대로 들어가게 된다.

세번째 for문

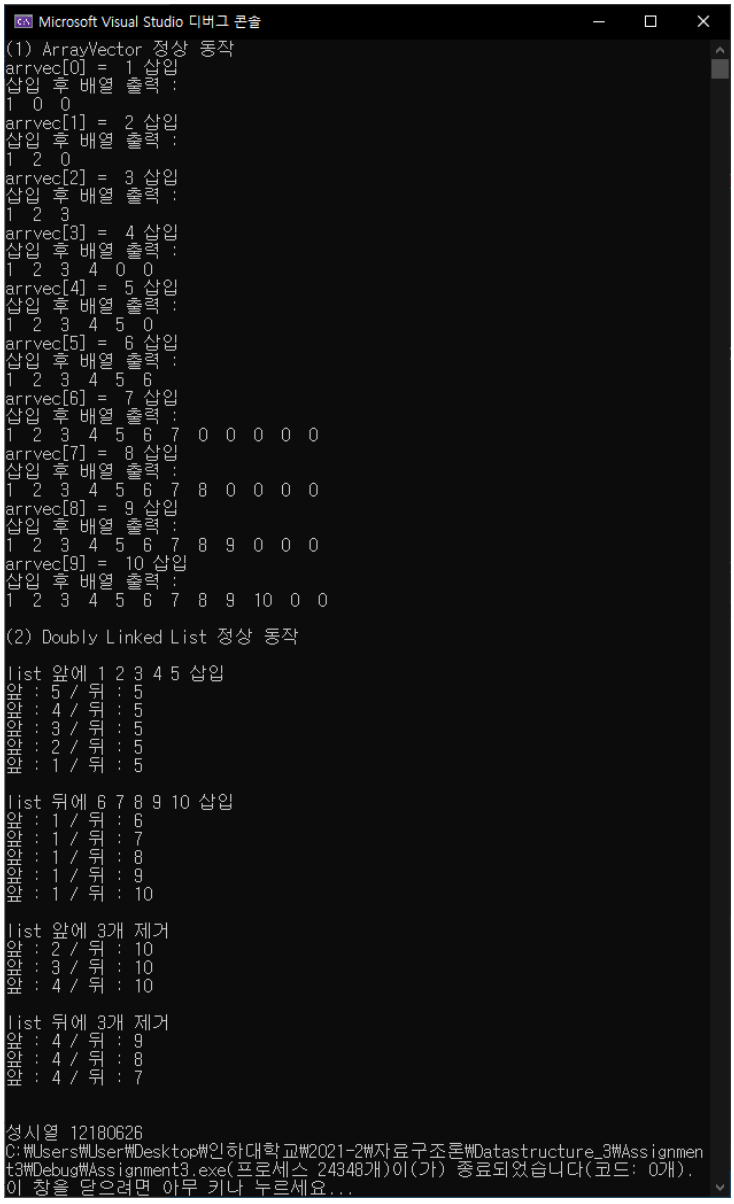
DDL의 앞에서부터 3개를 제거하여 1 2 3이 제거되고 남은 element는 4 5 6 7 8 9 10이 된다.

네번째 for문

DDL의 뒤에서부터 3개를 제거하여 10 9 8이 제거되고 남은 element는 4 5 6 7이 된다.

이 외의 코드에 대한 모든 설명은 주석으로 자세히 작성함.

[실행결과분석]



먼저 ArrayVector가 정상 동작하는지 보이기 위한 코드를 실행시켜 출력한다.

ArrayVector 클래스로 생성된 arrvec는 총 크기가 3칸이고 0으로 초기화되어있다.

이후 arrvec[0]에 1을 넣고 배열 전체를 출력하면 첫번째 칸에 1이 들어가 [1 0 0]임을 알 수 있다.

두번째는 arrvec[1]에 2를 넣고 배열 전체를 출력하면 두번째 칸에 2가 들어가 [1 2 0]임을 알 수 있다.

이렇게 삽입을 반복하여 진행하고, 용량과 element의 개수가 같아질 때 ArrayVector에 insert() 함수와 reserve()함수를 통해 용량을 두배로 늘린다. capacity=3이었기 때문에 6이 된다.

용량이 늘어난 뒤 이전과 같이 값 순서대로 삽입하고 element의 개수가 6이 될 때 다시 용량이 두배로 늘어나 12가 된다. 이를 arrvec[9] = 10까지 반복한 것이 출력결과이다.

다음은 Doubly Linked List가 정상 동작하는지 보이기 위한 코드를 실행시켜 출력한다.
DLinkedList 클래스로 생성된 DLL은 Doubly Linked List형태이다.
처음으로 list 앞을 통해 5 4 3 2 1 순으로 값을 삽입한다. 앞에서부터 삽입하기 때문에 가장 먼저 삽입한 값이 뒤로 가게 된다. 결국 5개의 값을 모두 삽입하면 1 2 3 4 5 순서대로 list에 저장된다.
두번째는 list 뒤를 통해 6 7 8 9 10 순으로 값을 삽입한다. 1 2 3 4 5가 이미 있는 list에 6 7 8 9 10을 차례대로 삽입한다. 두번째 for문이 끝난 뒤 1 2 3 4 5 6 7 8 9 10이 list에 저장된다.
세번째는 list 앞의 element를 3개 제거하는 것이다. 1 2 3 을 제거하고 4 5 6 7 8 9 10이 저장된다.
네번째는 list 뒤의 element를 3개 제거하는 것이다. 10 9 8을 제거하고 4 5 6 7이 저장된다.
마지막 출력문을 보면 앞이 4 뒤가 7인 것을 보아 4 5 6 7이 저장된 것이 맞다는 것을 확인할 수 있다.
이후 학번과 이름을 출력한다.

[결론 및 느낀점, 어려웠던 점]

자료구조론 세번째 과제를 하면서 ArrayVector와 Doubly Linked List에 대한 코드를 짜볼 수 있었다. 나에게는 수업이 조금 어렵게 느껴져 수업 때 제공해주신 강의노트를 많이 참고하여 코드를 작성할 수 있었다. 수업 중 교수님께서 설명해주실 때는 이해가 된다고 생각을 했는데 막상 직접 코드를 구성하려고 하니 정말 어려웠다. 그래도 ArrayVector와 Doubly Linked List에 대해서는 직접 코드를 짜보면서 이해를 하는데 도움이 되었다. 하지만 아쉽게도 문제 3번에 대한 코드를 구현하지 못했다. Stock Span 알고리즘에 대해 교수님께서 설명해주실 때는 정말 이해하기 쉽게 설명을 해주셔서 할 수 있을 것이라고 생각이 들었는데, 직접 구현을 하려니 구현에 어려움이 많았다. 중위표기식을 후위표기식으로 변환하는 알고리즘도 마찬가지로 교수님의 수업 때는 이해가 되었으니 실제로 구현을 하려니 어려움이 있었다. 또한, 각각 ArrayVector와 스택에 대해서 어느정도 이해를 하고 있는데 이 두개를 연결하는 것이 어려웠다. 스택에 대한 복습을 꾸준히 하고 이를 통해 ArrayVector를 이용하여 스택을 구현하는 방법도 반드시 연습해봐야겠는 생각을 했다.
이번 과제를 통해 ArrayVector와 Doubly Linked List에 대한 개념과 사용방법에 대해 확실히 이해할 수 있었다. 그리고 해결하지 못한 과제를 보며 복습의 중요성을 다시 한 번 더 느끼게 되었고, 다음 과제를 위해서 이전에 배웠던 개념들을 다시 한 번 복습하고 코드를 짜보는 시간을 가져야 겠다고 생각했다.