5주차 결과보고서

전공 : 컴퓨터공학 학년 : 2학년 학번 : 20151616 이름 : 최승환

아래에 보인 사항을 작성하여 다음 실험 시간에 제출하시오.

1. 실험 시간에 작성한 프로그램의 알고리즘과 자료구조를 요약하여 기술하시오.

자료구조

Array class

코드

#include "Array.h"

#include <cstdlib>

#include<iostream>

#include <cstdio>

using namespace std;

Array::Array(int size)

{

if(size<0)

{

printf("Array bound error\n");

}

else

{

data = new int[size];

len = size;

}

}

Array::~Array()

{

delete[] data;

}

int Array::length() const

{

return len;

}

int& Array::operator[](int i)

{

static int temp;

if((i >= 0) && (i < len))

{

return data[i];

}

else

{

printf("Array bound error\n");

return temp;

}

}

int Array::operator[](int i) const

{

if((i >= 0) && (i<len))

{

return data[i];

}

else

{

printf("Array bound error\n");

return 0;

}

}

void Array::print()

{

int cnt;

cout<<"[";

for(cnt=0;cnt<len;++cnt)

{

printf(" %d",data[cnt]);

}

cout<<" ]"<<endl;

}

생성자에서는 길이를 전달받는다. 만약 길이가 음수인 경우 에러메세지를 출력한다. 그리고 new를 이용해 data에 길이만큼의 int 배열을 만든다. 그리고 len에 문자열의 길이를 저장해준다. operator의 경우 int&를 반환해 배열을 반환할 수 있도록 한다. 또한 할당하고 싶은 배열의 길이에 문제가 있는 경우 리턴값도 고려해줘야 하기 때문에 임의의 변수 static int temp를 선언하고 에러메세지와 함께 이를 return 해준다.

RangeArray class

코드

#include <iostream>

using namespace std;

#include "RangeArray.h"

RangeArray::RangeArray(int a,int b):Array(b-a+1)

{

low = a;

high = b;

}

RangeArray::~RangeArray()

{

}

int RangeArray::baseValue()

{

return low;

}

int RangeArray::endValue()

{

return high;

}

int& RangeArray::operator [](int index)

{

return Array::operator [](index - low);

}

int RangeArray::operator [](int index) const

{

return Array::operator [](index - low);

}

Array를 상속받기 때문에 기본적인 기능은 모두 Array의 것을 따른다. 하지만 RangeArray는 음수의 인덱스로 시작하는 경우에도 배열을 할당해줄 수 있어야 한다. 이를 위해 index-low를 이용해 배열의 크기를 지정해준다.

알고리즘

Array class

생성자

우선 생성자에서는 사용자로부터 입력받는 사이즈가 0보다 작으면 에러메세지를 출력하도록 짜야한다. 만약 0보다 크거나 같으면 data에 공간을 할당하고 len에 size의 값을 넣는다.

소멸자

소멸자에서는 생성자에서 공간 할당한 메모리를 delete 해야한다.

int length()

길이 즉 len을 반환하면 된다. 여기서 size를 반환하면 안 된다.

int& operator[] (int i)

static tmp를 만들고 인덱스가 0보다 크거나 같고 len보다 작으면 data[i], 아니면 에러메세지와 tmp를 반환한다. tmp를 반환하는 이유는 함수의 형식을 맞추기 위해서이다.(int&를 반환해야 하므로)

int operator[] (int i)

const인덱스가 0보다 크거나 같고 len보다 작으면 data[i]를 반환하고, 아니면 에러메세지와 0을 반환한다.

void print()

앞 그리고 뒤에 [,]를 붙여서 출력하면 된다.

RangeArray Class

생성자

생성자에서는 Array 클래스를 상속했기 때문에 간단하다. low 값과 high 값에 입력받은 값들을 각각 넣고, :Array(high-low+1) 같은 방식으로 Array 클래스의 생성자를 사용하면 된다.

소멸자

Array 클래스와 다를게 없어서 그냥 괄호를 열고 닫으면 된다.

int baseValue()

입력받은 값 중에서 low에 저장한 값을 반환하면 된다.

int endValue()

입력받은 값 중에서 high에 저장한 값을 반환하면 된다.

int& operaotr[] (int i)

생성자와 같이 상속받은 Array의 함수를 사용해서 return Array::operator[] (i-low) 같은 방법으로 사용하면 된다.

int operator[] (int i) const

생성자와 같이 상속받은 Array의 함수를 사용해서 return Array::operator[] (i-low) 같은 방법으로 사용하면 된다.

2. 숙제 문제를 해결하기 위한 알고리즘 및 자료구조를 요약하여 기술하시오.

자료구조

Str Class

코드

#include <cstdio>

#include <iostream>

#include <cstring>

using namespace std;

//using namespace Str;

class Str{

private :

char \*str; // string의 내용.

int len; // string의 길이.

public :

Str(int leng);

Str(const char neyong[]);

~Str();

int length(void);

char\* contents(void);

int compare(class Str& a);

void operator=(const char a[]);

void operator=(class Str& a);

};

Str::Str(int leng)// leng 은 string의 길이.

{

if(leng<0)

{

cout << "size error!" << endl;

}

str = new char[leng];

len = leng;

}

Str::Str(const char neyong[])// neyong은 초기화할 내용이 들어감.~Str(); // 소멸자.

{

str = new char[strlen(neyong)];

strcpy(str,neyong);

len = strlen(neyong);

}

Str::~Str()// 소멸자.

{

delete str;

}

int Str::length(void)// string의 길이를 리턴하는 함수.

{

return len;

}

char\* Str::contents(void) // string의 내용을 리턴하는 함수.

{

return str;

}

int Str::compare(class Str& a) // a의 내용과 strcmp.

{

return strcmp(str,a.str);

}

void Str::operator=(const char a[]) // string의 값을 대입.

{

strcpy(str,a);

len = strlen(a);

return;

}

void Str::operator=(class Str& a) // Str의 내용을 대입.

{

strcpy(str,a.str);

len = a.len;

return;

}

private으로 두 개의 변수를 선언한다. 문자열을 저장하는 변수 str 와 문자열의 길이를 저장하는 변수 len이다. 하지만 이 str은 char\*형으로 선언되어 있기 때문에 생성자에서 문자열을 “I am a boy”와 같이 바로 받을 경우 문제가 생길 수 있다. 이 때문에 생성자에서는 char\*형을 받는 것이 아닌 const char 배열 형으로 받았다. 또, 생성자에서 해당 문자열을 저장할 때 str에 메모리가 할당되지 않은 상태로 받으면 세그먼트 오류가 발생할 수 있다. 이 때문에 str에 문자열을 저장하기 전에 new 를 이용해 문자열의 길이만큼 메모리를 할당해준다.

알고리즘

Str Class

생성자

int 형을 받았을 때뫄 문자열을 받았을 때로 나눠 만든다. int 형을 받은 경우 해당 길이만큼 문자열에 메모리를 할당해주고, len에 받은 길이를 저장한다. 만약 음수가 들어온 경우 에러 메세지를 출력해준다. 문자열을 받은 경우 문자열을 str에 복사해주고 길이를 재 len에 저장해준다.

소멸자

str을 delete한다.

int length()

길이를 return하도록 설정한다.

char\* contents()

str에 저장된 문자열을 return한다.

int compare(class Str a)

Str 클래스를 입력 받았을 경우에는 strcmp(str,a.str) 같은 방법으로 문자열을 비교하고 그 값을 return한다.

int compare(const char a)

파라미터로 문자열을 입력 받았을 경우에는 strcmp(str,a.str)으로 문자열을 비교하고 그 값을 return한다.

void operator=(const char a[])

넣을 문자열의 길이만큼 len에 저장하고 기존의 strcpy를 이용해 str에 a를 복사해준다.

void operator=(class Str& a)

strcpy(str,a.str)을 이용해 객체 a의 문자열을 복사해준다. 문자열의 길이도 a에 저장된 문자열의 길이로 만들어준다.