2020년도 2학기 컴퓨터공학설계및실험I

3주차 결과 보고서

20170175 김태안

1. 실습 목적

C++는 객체지향 프로그래밍 개념을 도입하여 클래스(Class)를 확장하고 수정하는 수단으로 클래스 상속(Class Inheritance)을 제공한다. 이번 실험에서는 효율적인 프로그래밍을 위하여 클래스의 상속 방법을 익혀보도록 한다.

1. 관련 이론

**객체 지향 프로그래밍(Object-Oriented Programming, OOP)**

많은 프로그래밍 언어가 개발되면서 언어는 패러다임을 갖게 되었다. 그 중에서 객체 지향 프로그램은 프로그램을 상호 작용하는 객체의 집합으로 보는 관점이다. 각 객체는 class의 instance로 구성되어 있고, class는 서로 상속 관계에 있는 계층 구조이다. 객체 지향 프로그래밍의 특징은 1) 현실 세계의 문제를 잘 반영하는 모델을 구현할 수 있고, 2) 모듈화되어 있으며, 3) 소프트웨어의 재사용이 가능하고, 4) 각 객체가 병렬적으로 실행될 수 있다.

객체 지향 프로그래밍을 구현하기 위해 많은 기법을 사용하는데, 그 중 하나가 추상 데이터 타입 (Abstract Data Type, ADT)이다. ADT는 캡슐화와 정보 은닉을 지원하는 데이터 타입이다. 캡슐화란 데이터와 코드를 하나로 묶어 하나의 데이터로 정의되는 것이다. 정보 은닉은 객체의 표현과 연산자를 숨김으로써 외부의 접근을 막는 것을 의미한다. 이러한 특징은 소프트웨어 구성 요소들의 상호 의존성을 떨어트려 소프트웨어 신뢰도를 높이고 수정 가능성을 열어 둔다. (낭종호 5-9)

**객체 (Objects)와 Class, Instance**

객체 지향 프로그래밍의 객체란, 고유한 데이터와 method를 갖는 entity이다. 객체는 고유 데이터를 가지고, 고유 데이터를 다루기 위한 명령(method)로 이루어진다. 이때, 고유 데이터를 멤버 변수, 명령을 멤버 함수라고 한다. (낭종호 12)

Class는 객체의 틀이다. ‘홍길동’이라는 객체가 학번, 이름, 나이라는 정보를 가지고 있다고 할 때, 홍길동만을 위한 객체를 생성하는 대신 학번, 이름, 나이를 가지고 있는 틀 ‘학생’이라는 Class를 생성해 사용할 수 있다. 객체는 데이터와 method, 그리고 상속받은 부모 Class를 담고 있다. 이때, 홍길동을 저장하기 위해 생성한 객체 학생 1을 Instance라고 한다. (낭종호 16-17)

**상속 (Inheritance)**

Class간의 상속을 통해 더 편리하게 Class를 구현할 수 있다. 한 Class는 다른 Class를 상속할 수 있다. 이때 기반이 되는 Class를 부모, 상속받는 Class를 자식이라고 한다. 상속을 받게 되면 자식 Class는 부모 Class에 자식 Class 고유의 데이터와 method를 더한 구조로 이루어진다. 따라서 자식 Class는 부모 Class의 데이터와 method를 모두 사용할 수 있다. 상속을 사용하면 이미 사용한 코드를 재활용해 구현이 단순해지고, 일상의 문제를 프로그램으로 더 쉽게 표현할 수 있다. (낭종호 18-24)

1. 실습 구현 내용

RangeArray는 기존의 Array와는 달리 Index의 시작을 사용자가 임의로 지정할 수 있는 Array이다. 기존의 Array를 Class로 구현한 뒤, 이를 상속받아 RangeArray로 구현한다.

1. **Array의 자료구조**

|  |
| --- |
| #include <iostream>  #include "Array.h"  using namespace std;  Array :: Array(int size){  if (size <= 0){  cout<<"Error: Size Smaller than 1"<<endl;  }  else {  data = new int[size];  len = size;  }  }  Array :: ~Array() {  delete []data;  }  int Array :: length() const {  return len;  }  int & Array :: operator [] (int i) {  static int tmp;  if ((i < len)&&(0 <= i)){  return data[i];  }  else {  cout<<"Array Bound Error!"<<endl;  return tmp;  }  }  int Array :: operator [] (int i) const {  static int tmp;  if ((i < len)&&(0 <= i)){  return data[i];  }  else {  cout<<"Array Bound Error!"<<endl;  return 0;  }  }  void Array :: print (void) {  int j;  cout<<"[";  for (j=0; j<len; j++) {  cout<<" "<<data[j];  }  cout<<"]"<<endl;  } |

Array는 정수의 배열이다. 숫자를 입력 받고 그 크기와 크기만큼의 배열을 저장한다. Array Class는 아래의 멤버함수를 가진다.

**Array(int size);**

Array를 생성한다. size를 len에 저장하고 배열을 할당한다.

**~Array();**

생성된 Array의 메모리를 해제하여 삭제한다.

**int length() const;**

Array의 길이를 반환한다.

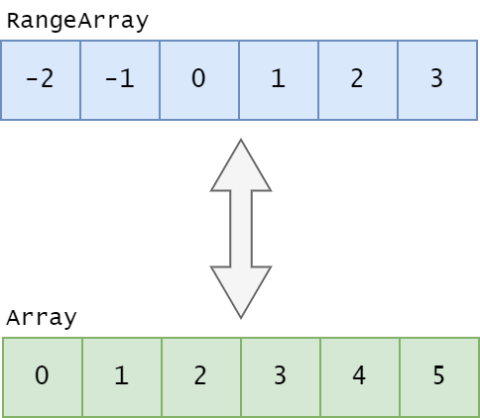
**int & operator [] (int i);, int operator [] (int i) const;**

Array가 []와 함께 사용될 때, i번째 원소를 반환한다.

**void print();**

Array의 모든 내용을 출력한다.

1. **RangeArray의 알고리즘**



i-base

RangeArray를 구현하기 위해 실제로 인덱스가 임의의 숫자인 배열을 만드는 대신, 같은 크기의 배열을 만들고 인덱스를 입력할 때마다 변환해 마치 인덱스가 다른 범위를 가지는 것처럼 표현할 수 있다. 예를 들어, 인덱스가 -2~3인 RangeArray를 구현해보자. 실제로는 크기가 5인 배열을 생성한다. RangeArray[1]의 값을 구하기 위해서는 1을 입력받으면 이를 Array에 맞는 숫자인 3으로 변환해 그 값을 가지고 온다. 이때, 실제 배열의 인덱스는 입력값 i – RangeArray의 시작값 base이다. 즉, RangeArray의 모든 인덱스 탐색에 대해, Array[i-base]를 통해 값을 반환한다면 사용자가 RangeArray 기준으로 원하던 값을 반환할 수 있다.

1. **RangeArray의 자료구조**

|  |
| --- |
| #include "RangeArray.h"  #include <iostream>  using namespace std;  RangeArray :: RangeArray (int start, int finish) : Array (finish-start +1) {  base = start;  end = finish;  }  RangeArray :: ~RangeArray (void){    }  int & RangeArray :: operator [] (int i) {  Array:: operator [] (i-base);  }  int RangeArray :: operator [] (int i) const {  Array::operator [](i-base);  }  int RangeArray :: baseValue (void) {  return base;  }  int RangeArray :: endValue (void) {  return end;  } |

RangeArray는 Array Class를 상속받는다. base와 end를 입력받아 저장하고, 인덱스를 사용하는 모든 연산자는 i-base의 값을 넣은 Array의 연산자 오버로딩을 그대로 사용한다. RangeArray는 추가로 base와 end를 반환하는 함수 int RangeArray::baseValue(void)와 int RangeArray::endValue(void)를 가진다.

1. 실습 환경

cspro.sogang.ac.kr

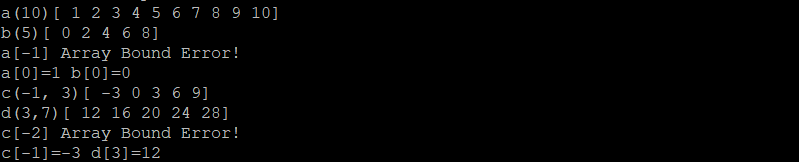
OS: Ubuntu 16.04.2 LTS (GNU/Linux 4/4/0-184-generic x86\_64)

1. 실습 결과 및 분석

구현한 Array Class와 RangeArray Class의 동작을 확인하기 위해 아래와 같은 main 함수를 구현하였다.

|  |
| --- |
| int main () {  int i, x, y;  Array a(10), b(5);  for(i=0; i<a.length(); i++) a[i] = i + 1; // left value  for(i=0; i<b.length(); i++) b[i] = i \* 2;  cout<<"a(10)"; a.print();  cout<<"b(5)"; b.print();  cout<<"a[-1] ";  a[-1]=7;  x = a[0];  y = b[0];  cout<<"a[0]="<<x<<" b[0]="<<y<<endl;  RangeArray c(-1, 3), d(3,7);  for(i=c.baseValue(); i<=c.endValue(); i++) c[i] = i \* 3; // left value  for(i=d.baseValue(); i<=d.endValue(); i++) d[i] = i \* 4;  cout<<"c(-1, 3)"; c.print();  cout<<"d(3,7)"; d.print();  cout<<"c[-2] ";  c[-2] = 3;  x = c[-1]; y = d[3];  cout<<"c[-1]="<<x<<" d[3]="<<y<<endl;  return 0;  } |

메인 함수의 실행 결과는 다음과 같다.



a(10)을 실행하면 크기가 10인 Array가 만들어지며, for문을 통해 1, 2, …,10을 넣는다. b(5)를 실행하면 크기가 5인 Array가 만들어지며, for문을 통해 0, 2,…, 8을 넣는다. print() 함수를 통해 a와 b의 내용을 확인할 수 있다. a[-1]은 범위를 벗어난 인덱스이므로 에러 메시지를 호출한다. [] 연산자를 사용하여 a[0]과 b[0]의 값을 x, y에 저장한다. c(-1, 3), d(3, 7)을 실행하면 각각 -1부터 3까지, 3부터 7까지의 인덱스를 가지는 RangeArray를 생성한다. for문을 통해 c는 -3, 0, …, 9를, d는 12, 16, …, 28을 입력한다. 마찬가지로 print() 함수를 통해 배열의 내용을 확인할 수 있다. c[-2]는 범위를 벗어난 인덱스이므로 에러 메시지를 호출하며, [] 연산자를 사용해 값을 x, y에 저장하고 출력한다.

1. 과제

**str Class**

str Class는 문자열을 다루는 Class이다. 문자열을 입력받아 길이와 내용을 저장하고, 다른 문자열과 비교한다. = 연산자를 통해 다른 문자열을 대입하거나, 반환할 수 있다.

|  |
| --- |
| #include "str.h"  #include <iostream>  #include <string.h>  #include <stdio.h>  using namespace std;  Str :: Str(int leng){  len = leng;  }  Str :: Str(char \*neyong){  len = strlen(neyong);  str = new char [len+1];  strcpy(str, neyong);  }  Str :: ~Str(void) {  delete[] str;  return;  }  int Str :: length(void) {  return len;  }  char\* Str :: contents(void) {  return str;  }  int Str :: compare(class Str& a){  return strcmp(str, a.str);  }  int Str :: compare(char \*a){  return strcmp(str, a);  }  void Str :: operator = (char \*a){  delete[] str;  len = strlen(a);  str = new char [len+1];  strcpy(str, a);  }  void Str :: operator = (class Str& a) {  delete[] str;  len = strlen(a.str);  str = new char [len+1];  strcpy(str, a.str);  } |

str Class는 다음의 멤버 함수를 가진다.

**Str(int leng);**

문자열의 길이를 입력 받아 저장한다.

**Str(char \*neyong);**

문자열의 내용을 입력 받아 내용과 길이를 저장한다.

**~Str();**

저장한 문자열을 제거한다. (메모리 해제)

**int length(void);**

저장한 문자열의 길이를 반환한다.

**char \*contents(void);**

저장한 문자열을 반환한다.

**int compare(class Str& a);, int compare(char \*a);**

저장한 문자열과 a를 비교한다. strcmp와 같은 기능을 한다.

**void operator=(char \*a);**

문자열을 str에 대입한다.

**void operator=(class Str& a);**

str을 문자열에 대입한다.

1. 결론

이번 실험에서는 C++의 기본 개념과 문법에 대해 알아보았다. C++은 C의 확장으로 객체 지향 프로그래밍을 위한 상속과 Class 개념이 추가되었다. Class란 객체의 틀로, 객체가 가지고 있어야하는 변수와 객체를 효율적으로 사용하기 위한 멤버 함수를 가지고 있다. 새로운 자료형과 그 기능을 정의하는 것과 비슷하다.

상속은 한 Class가 다른 Class를 가져와 사용하는 방법이다. 부모 Class의 구성요소와 함수를 그대로 사용하면서 추가적인 데이터와 함수를 가지는 새 Class를 생성한다. 이를 통해 코드의 반복을 줄일 수 있다.

1. 참고 문헌

낭종호 교수님. ”객체지향 프로그래밍.” *프로그래밍 언어 강의자료*. 서강대학교 공과대학 컴퓨터공학과, 2020, pp.5-9.