A-1

Report

제목 : C++프로그래밍

과제 6번

과제내용

p350~352 연습문제 1~7, 11~16번

p353 프로그래밍 과제 2번

미수행과제내역

담당교수 : 배성현 교수님

학과명 : 생명과학과

학번 : 5355002

이름 : 정유진

**일반 연습문제 과제**

350p

1. Point라는 클래스가 다음과 같이 정의되어 있다고 하자. 다음 코드 중에서오류를 찾아서 올바르게 수정하라.

#include <iostream>

using namespace std;

class Point {

int x, y;

public:

Point(int x, int y) : x(x), y(y) { }

void setX(int x) { this->x = x; }

void setY(int y) { this->y = y; }

int getX() { return x; }

int getY() { return y; }

};

int main() {

Point \*p = new Point(100, 200);

p.setX(30); **// p->setX(30);**

p.setY(60); **// p->setY(60);**

delete p;

return 0;

}

p는 포인터 변수이기 때문에 -> 연산자를 이용하여 함수를 불러와야합니다**.**

2. 1번 코드를 스마트 포인터의 일종인 unique\_ptr을 사용하여 다시 작성하라.

#include <iostream>

using namespace std;

class Point {

int x, y;

public:

Point(int x, int y) : x(x), y(y) { }

~Point(){}

void setX(int x) { this->x = x; }

void setY(int y) { this->y = y; }

int getX() { return x; }

int getY() { return y; }

};

int main() {

**unique\_ptr<Point> p(new Point(100, 200));** // 포인터변수사용

p->setX(30);

p->setY(60);

return 0;

}

3. 1번 코드에서 100개의 Point 객체를 저장할 수 있는 동적 객체 배열을 생성하고 Point 객체의 x, y값을 난수로 채워보자.

int main() {

srand(time(NULL));

Point\* p = new Point[100]; // 동적 객체 배열 생성

for (int i = 0; i < 100; i++) {

p[i].setX(rand() % 100);

p[i].setY(rand() % 100);

}

return 0;

}

4. 1번 코드의 Point 클래스의 멤버 함수에 최대한 많이 const를 추가하여 보라.

Point(int x, int y) : x(x), y(y) { }

void setX(int x) { this->x = x; }

void setY(int y) { this->y = y; }

int getX() **const** { return x; }

int getY() **const** { return y; }

5. 1번 코드의 Point 클래스의 멤버 함수에 최대한 많이 this를 추가하여 보라.

Point(int x, int y) : x(x), y(y) { }

void setX(int x) { **this->**x = x; }

void setY(int y) { **this->**y = y; }

int getX() { return **this->**x; }

int getY() { return **this->**y; }

351p

6. 2개의 Point 객체가 동일한지를 검사하는 다음과 같은 함수를 정의한다고 하자. “값에 의한 호출”을 사용하는 것과 “참조에 의한 호출”을 사용하는 것을 효율적 측면에서 비교하여 보자. 어떤 방법이 효율적인가?

bool isEqual(Point p1, Point p2){

return ((p1.getX() == p2.getX()) && (p1.getY() == p2.getY()));

}

bool isEqual(Point \*p1, Point \*p2){

return ((p1->getX() == p2->getX()) && (p1->getY() == p2->getY()));

}

**해답 :** 값에 의한 호출일시에 매개변수가 전달받은 인자로 초기화가 되며 매개변수의 크기만큼 공간이 할당됩니다. 해당문장에서는 class의 객체를 매개변수로 두었기 때문에 2개의 변수와 5개의 메소드로 이루어진 매개변수의 공간이 할당되었습니다. 하지만 \*연산자를 이용하여 포인터변수를 사용할시에는 Point \*p1과 Point \*p2는 Point 객체의 주소값을 가르키고 포인터변수는 4바이트의 공간을 할당받기 때문에 데이터 공간을 차지하는 비중을 낮출 수 있습니다. 그러므로 참조에 의한 호출이 좀더 효율적입니다.

7. 6번 문제의 isEqual () 함수를 참조자로 다시 작성해보자. 효율성면에서 분석하여 보자.

bool isEqual(Point& p1, Point& p2) {}

**bool isEqual(Point& p1, Point& p2){**

**return ((p1.getX() == p2.getX()) && (p1.getY() == p2.getY()));**

**}**

**해답 :** isEqual함수의 매개변수를 참조자를 이용하여 작성한다면 별도의 매개변수의 데이터 공간이 생성되는것이 아니라 기존에 존재하는 변수에 매개변수의 이름이 붙는것이기 때문에 데이터 공간을 효율적으로 관리할 수 있습니다.

352p

11. 사각형을 나타내는 클래스 Rect를 가지고 실습하여 보자. Rect 클래스에 접근자와 설정자, 사각형의 면적을 계산하는 getArea()와 둘레를 계산하는 getPerimeter()를 추가하여 보자. 기타 print()와 같은 필요한 멤버 함수도 추가한다. const를 추가할 수 있으면 최대한 추가하라.

#include <iostream>

#include <string>

using namespace std;

class Rect{

int width, height;

public:

**int getArea() const { return (width \* height); } // 사각형의 면적 계산**

**int getPerimeter() const { return ((width \* 2) + (height \* 2)); } // 둘레계산**

**void print() const {**

**cout << "사각형의 면적 : " << getArea() << endl;**

**cout << "사각형의 둘레 : " << getPerimeter() << endl;**

**}**

};

int main(){

Rect \*p = \_\_\_\_\_\_\_\_\_\_\_\_; //①

return 0;

}

12. 11번 코드에 필요한 생성자를 추가하여 보자.

**Rect(int width, int height){**

**this->width = width;**

**this->height = height;**

**}**

13. ① 위치에 동적으로 객체를 생성하여 보자. 폭과 높이를 10과 20으로 설정하는 생성자를 호출한다. 동적 객체를 삭제하는 코드도 추가한다.

**int main() {**

**Rect \*p = new Rect(10, 20);//①, 동적 객체 생성**

**delete p; // 동적객체 삭제코드**

**return 0;**

**}**

14. p를 통하여 getArea()를 호출하여 보자.

**int rectArea = p->getArea();**

15. Rect 객체 두개를 받아서 같은 크기인지를 체크하는 일반 함수 is\_equal()을 작성하여 보자. 다음과 같은 함수 원형을 가정한다.

bool is\_equal(Rect r1, Rect r2);

**bool is\_equal(Rect r1, Rect r2) {**

**return (r1.getArea() == r2.getArea());**

**}**

16. 15번 함수 is\_equal()을 약간 수정하여 보자. 이번에는 함수의 매개 변수를 참조자로 선언하여 보자. 수정되어야 하는 코드가 있는가? 15번과의 차이점은 무엇일까? 실행속도는 어떤 것이 빠를까? 위험성은 없는가?

bool is\_equal(Rect& c1, Rect&c2);

**bool is\_equal(Rect &c1, Rect &c2) {**

**return (c1.getArea() == c2.getArea());**

**}**

r1과 r2의 데이터 공간이 생성되는 15번 문제와는 달리 전달된 변수에 c1, c2라는 이름이 붙어서 데이터 공간을 효율적으로 관리할 수 있고 속도또한 빠릅니다. 그러나 별도의 데이터 공간을 가진 r1, r2와는 달리 c1, c2는 변수의 값을 변경할 수 있는 여지가 있기 때문에 주의가 필요합니다.

**프로그래밍 과제**

353p

2. 사용자가 몇 개의 이름을 입력할 것인지를 물은 후에 동적 배열을 생성하여 사용자로부터 받은 이름을 저장하는 프로그램을 작성하라. new를 이용하여 string의 동적 배열을 생성한다. 이 동적 배열에 이름들을 저장한다.

1) 소스코드

#include <iostream>

#include <string>

using namespace std;

int main() {

int num;

cout << "얼마나 많은 이름을 입력 하시겠습니까? ";

cin >> num;

string \*arr = new string[num]; // 동적배열 생성

for (int i = 0; i < num; i++) {

cout << "이름 입력 # " << i+1 << ": ";

cin >> arr[i]; // 기존 배열 입력방식과 동일

}

cout << endl;

cout << "다음은 이름 목록입니다" << endl;

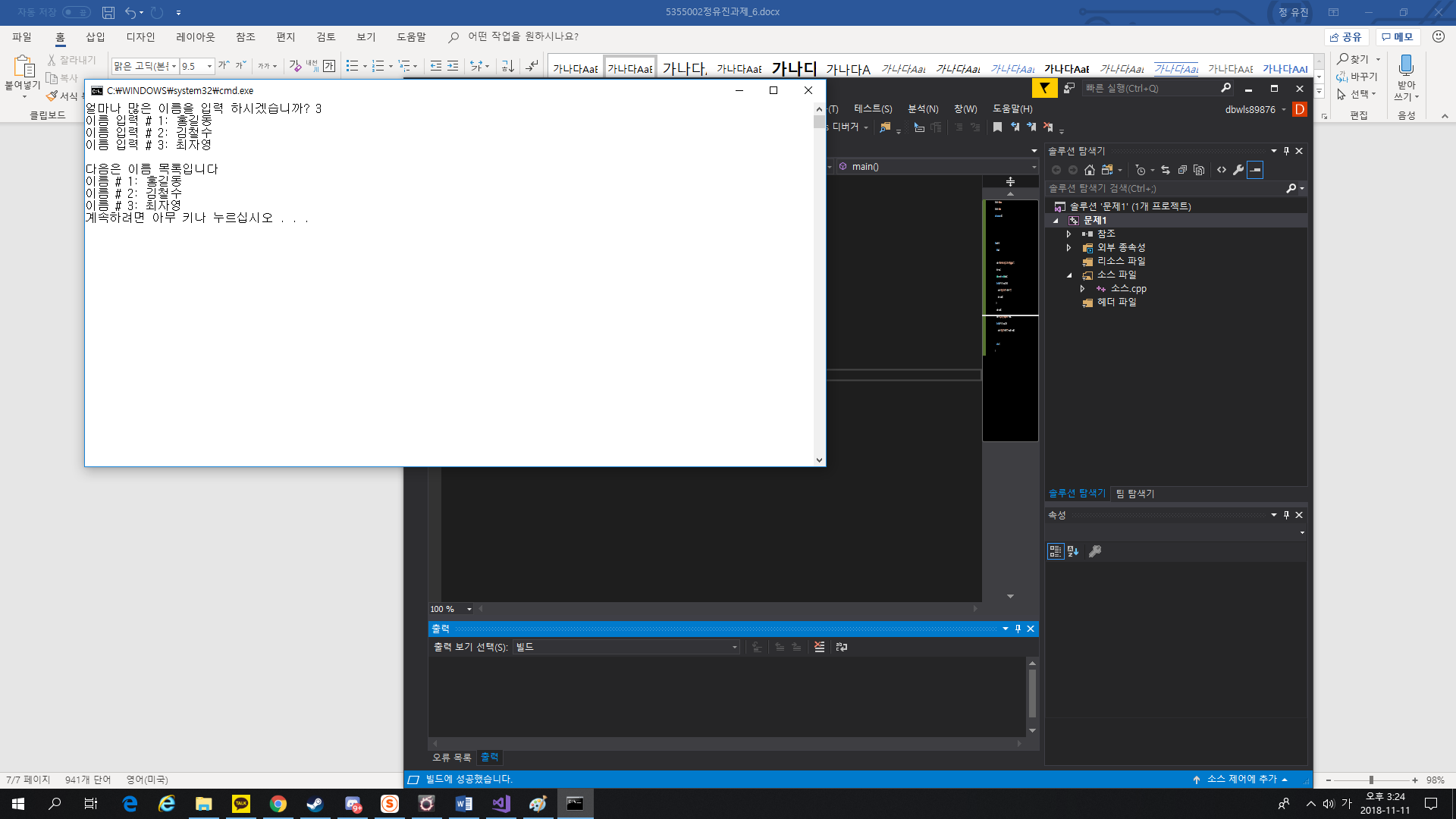
for (int i = 0; i < num; i++)

cout << "이름 # "<<i+1<<": "<< arr[i] << endl; // 배열에 담긴 문자 출력

return 0;

}

2) 출력결과



3) 소스코드 설명

사용자로부터 이름을 입력할 횟수를 입력받고 횟수만큼 입력을 입력한 뒤에 입력받은 이름들을 출력해주는 프로그램입니다. 입력받은 크기만큼 동적 배열을 선언하여 데이터공간을 할당받은 뒤에 기존의 배열의 입출력 방식과 동일하게 입, 출력 하였습니다.

4) 의견 및 고찰

14번 문제를 푸는 중간에 C3867 코드로 인해 고생을 했다. 일반 연습 문제를 풀면서 혹시라도 실수를 하지 않기 위해 IDE를 이용하여 오류가 없는지 확인하고 문제를 적는데 14번 문제를 푸는 과정에서 return (r1.getArea == r2.getArea); 가 비표준 구문이라고 &를 사용하여 멤버 포인터를 만들어라는 오류가 자꾸 떠서 한참을 해맸다. 30분정도 어디서 문제가 뜬건지 알아보다가 검색포털에서 검색을 해봤는데 함수를 불러오면서 ()를 입력하지 않았기 때문에 오류가 발생한 것이었다. 이를 깨닫고 오류는 사소한 실수로부터 온다는 사실을 뼈저리게 다시 절감하였다. 그외에는 문제를 푸는데 큰 어려움이 없었으나 스마트 포인터를 사용하는게 아직 어색해 책을통해 복습하는 시간을 한번 더 가졌다.