**REPORT**

**[실습 과제 4주차]**



**과 목 : 심화프로그래밍02**

**담당교수 : 윤성림 교수님**

**학 과 : 컴퓨터공학과**

**학 번 : 2021111971**

**이 름 : 이재혁**

**제 출 일 : 2024.3.31**

텍스트, 클립아트이(가) 표시된 사진

자동 생성된 설명

**실습과제 1번**

**텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명**

**문제분석**

생성자를 여러 개 선언할 수 있는지 확인하는 문제입니다.

**프로그램 설계**

생성자를 여러 개 선언해 매개변수가 있을 때, 없을 때 다르게 객체가 생성되도록 합니다.

**소스코드 및 주석**

// 2024.03.28

#include <iostream>

using namespace std;

class Tower {

public:

Tower() { // 매개변수가 없을 때는 기본 1로 설정

height = 1;

}

Tower(int h) { // int형 매개변수가 있을 때 그 값으로 설정

height = h;

}

int getHeight(); // height 변수는 class외부에서 접근할 수 없기 때문에, method를 통해 return;

private:

int height;

}; // 높이 값을 가지는 Tower class

int Tower::getHeight() {

return height;

}

int main() {

Tower myTower;

Tower seoulTower(100);

cout << "높이는 " << myTower.getHeight() << "미터" << endl;

cout << "높이는 " << seoulTower.getHeight() << "미터" << endl;

}

**결과 및 결과분석**

****

**소감**

배열을 선언하고 초기화하지 않으면 쓰레기 값이 들어가는 경우가 있는데, 기본생성자로 초기화를 해주기 때문에 쓰레기 값 걱정을 하지 않아도 돼서 좋은 것 같습니다.

**실습과제 2**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**문제분석**

객체 함수를 생성해 private값에 접근할 수 있는지 확인하는 문제입니다.

**프로그램 설계**

문자열로 년월일 이 한번에 들어오는 경우 각각 나눠서 변수에 저장하는 생성자를 추가로 선언합니다.

객체 변수를 return하는 객체 함수를 선언합니다.

**소스코드 및 주석**

#include <iostream>

#include <string>

using namespace std;

class Date {

public:

Date(int y, int m, int d) {

year = y;

month = m;

day = d;

}

Date(string s) {

year = stoi(s.substr(0, 4)); // s[0] 부터 4개의 값을 문자열로 return 후 int로 변환

month = stoi(s.substr(5, 1)); // s[5] 부터 1개의 값을 문자열로 return 후 int로 변환

day = stoi(s.substr(7)); // s[7] 부터 끝까지 문자열로 return 후 int로 변환

}

void show() {

cout << year << "년" << month << "월" << day << "일" << endl;

} // class가 가지고 있는 값들을 출력

int getYear() {

return year;

}

int getMonth() {

return month;

}

int getDay() {

return day;

}

// 년 원 일 을 반환하는 함수들

private:

int year, month, day;

};

int main() {

Date birth(2014, 03, 20);

Date independenceDay("1945/8/15");

independenceDay.show();

cout << birth.getYear() << "," << birth.getMonth() << "," << birth.getDay() << endl;

}

**결과 및 결과분석**



**소감**

객체 변수를 초기화 할 때 private접근 지정자로 선언을 했기 때문에,

생성자나 객체 변수의 균일한 방식으로 객체 변수를 설정할 수 있어 사람의 실수를 줄일 수 있다고 느꼈습니다.

**실습과제 4**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**문제분석**

멤버 변수에 접근에 각각 다른 작업을 할 수 있는 객체함수를 선언할 수 있는지 확인하는 문제입니다.

**프로그램 설계**

각각 멤버 변수를 초기화 하고 멤버 변수를 바꿀 수 있는 객체 함수를 작성합니다.

**소스코드 및 주석**

#include <iostream>

using namespace std;

class CoffeeMachine

{

public:

CoffeeMachine(int a, int b, int c)

{

coffee = a;

water = b;

sugar = c;

} // 커피량:a, 물량:b, 설탕:c로 초기화

void drinkespresso()

{

coffee -= 1;

water -= 1;

}; // 커피 1, 물 1 소비

void drinkAmericano()

{

coffee -= 1;

water -= 2;

} // 커피 1, 물 2 소비

void drinksugarCoffee()

{

coffee -= 1;

water -= 2;

sugar -= 1;

} // 커피 1, 물 2, 설탕 1 소비

void fill()

{

coffee = 10;

water = 10;

sugar = 10;

} // 커피 10, 물 10, 설탕 10으로 채우기

void show()

{

cout << "커피 머신 상태, " << "커피: " << coffee << " \t물: " << water << " \t설탕: " << sugar << endl;

} // 현재 커피 머신의 상태 출력

private: // 커피 물 설탕의 양을 저장할 멤버변수선언

int coffee;

int water;

int sugar;

};

int main()

{

CoffeeMachine java(5, 10, 3);

java.drinkespresso();

java.show();

java.drinkAmericano();

java.show();

java.drinksugarCoffee();

java.show();

java.fill();

java.show();

}

**결과 및 결과분석**

텍스트, 폰트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명

**소감**

객체 함수는 늘 균일한 작업을 하기 때문에, 실수로 멤버 변수를 잘못 수정하는 일을 줄일 수 있다고 생각했습니다.

**실습과제 6**

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**문제분석**

클래스 내부에 적절한 난수 발생 함수를 작성할 수 있는지 확인하는 문제입니다.

**프로그램 설계**

do-while문을 사용해 난수가 짝수가 아닌 동안 계속 난수를 발생시킵니다.

범위가 정해진 경우 범위 내에서 짝수가 아닌 동안 계속 난수를 발생시킵니다.

**소스코드 및 주석**

#include <iostream>

#include <cstdlib>

#include <ctype.h>

#define MAX 32768 // 난수의 최댓값

using namespace std;

class EvenRandom {

public:

int next() {

do {

even = rand() % MAX;

} while (even % 2 != 0); // 먼저 난수를 생성하고 짝수가 아니면 다시 반복

return even;

}

int nextInRange(int a, int b) {

do {

even = rand() % (b - a + 1) + a;

} while (even % 2 != 0);

return even; // 숫자범위 내에서 난수를 생성하고 짝수가 아니면 다시 반복

}

private: // 난수가 짝수 값이 맞는지 저장해서 확인할 멤버변수 설정

int even;

};

int main() {

EvenRandom r;

srand((unsigned)time(0));

cout << "\n------------ 0에서 " << MAX << "까지의 랜덤 짝수 10 개 ------------" << endl;

for (int i = 0; i < 10; i++) {

int n = r.next();

cout << n << ' ';

}

cout << endl

<< endl

<< "-- 2에서 " << "10 까지의 짝수 정수 10--" << endl;

for (int i = 0; i < 10; i++) {

int n = r.nextInRange(2, 10); // 2에서 10 사이의 랜덤한 짝수

cout << n << ' ';

}

cout << endl;

}

**결과 및 결과분석**

텍스트, 폰트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명

모든 숫자가 짝수로 출력 된 것을 확인할 수 있습니다.

**실습과제 7**

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**문제분석**

홀수, 짝수의 경우를 선택하는 조건을 만들 수 있는지 확인하는 문제입니다.

**프로그램 설계**

사용자로부터 입력을 받아, 0일때는 짝수, 1일 때는 홀수인 난수들을 발생시키도록 합니다.

**소스코드 및 주석**

#include <iostream>

#include <cstdlib>

#include <ctype.h>

#define MAX 32768 // 난수의 최댓값

using namespace std;

class SelectableRandom {

public:

SelectableRandom() {

cout << "짝수 : 0, 홀수 : 1 입력" << endl;

cin >> select;

} // SelectableRandom 객체가 생성될 때 0 또는 1을 입력 받아 select 멤버 변수에 저장한다.

int next() {

if (select == 0) {

do {

random = rand() % MAX;

} while (random % 2 != 0);

}

else if (select == 1) {

do {

random = rand() % MAX;

} while (random % 2 == 0);

}

// select가 0 이면 짝수가 될 때 까지 난수 생성

// select가 1 이면 홀수가 될 때 까지 난수 생성

return random;

}

int nextInRange(int a, int b) {

if (select == 0) {

do {

random = rand() % (b - a + 1) + a;

} while (random % 2 != 0);

}

else if (select == 1) {

do {

random = rand() % (b - a + 1) + a;

} while (random % 2 == 0);

}

// select가 0 이면 범위내에서 짝수가 될 때 까지 난수 생성

// select가 1 이면 범위내에서 홀수가 될 때 까지 난수 생성

return random;

}

private:

int random; // 난수값이 저장 될 변수

int select;

};

int main() {

SelectableRandom r;

srand((unsigned)time(0));

cout << "\n----------- 0에서 " << MAX << "까지의 랜덤 정수 10 개 -----------" << endl;

for (int i = 0; i < 10; i++) {

int n = r.next(); // 0에서 MAX(32767) 사이의 랜덤한 정수

cout << n << ' ';

}

cout << endl

<< endl

<< "-- 2에서 9 까지의 랜덤 정수 10--" << endl;

for (int i = 0; i < 10; i++) {

int n = r.nextInRange(2, 9); // 2에서 9 사이의 랜덤한 정수

cout << n << ' ';

}

cout << endl;

}

**결과 및 결과분석**

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명

**실습과제 8**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**문제분석**

자동 인라인 함수를 사용할 수 있는지 확인하는 문제입니다.

**프로그램 설계**

클래스 선언부에는 함수의 원형만 선언하고 클래스 외부에 함수를 구현하면 컴파일러가 자동으로 인라인 처리를 해줍니다.

**소스코드 및 주석**

#include <iostream>

#include <string>

using namespace std;

class Integer {

public:

Integer(int);

// 객체가 생성될 때 정수형을 매개변수로 받을 경우 num에 저장한다.

Integer(string);

// 객체가 생성될 때 문자열을 매개변수로 받을 경우 int로 바꾼후 저장한다

int get();

// 객체내부의 num의 값을 반환한다.

void set(int);

// 매개변수로 입력받은 정수로 num의 값을 설정한다.

void set(string s);

// 매개변수로 입력받은 문자열을 num의 값으로 설정한다.

int isEven();

// num에 저장되어 있는 값이 짝수면 1 그렇지 않으면 0을 반환한다.

private:

int num;

};

// 클래스 선언부에 구현된 함수는 컴파일러가 자동으로 인라인 처리를 해준다.

Integer::Integer(int n) {

num = n;

}

Integer::Integer(string s) {

num = stoi(s);

}

int Integer::get() {

return num;

}

void Integer::set(int n) {

num = n;

}

void Integer::set(string s) {

num = stoi(s);

}

int Integer::isEven() {

if (num % 2 == 0) {

return 1;

}

else return 0;

}

int main() {

Integer n(30);

cout << n.get() << ' '; // 30 출력

n.set(50);

cout << n.get() << ' '; // 50 출력

Integer m("300");

cout << m.get() << ' '; // 300 출력

n.set("500"); // 500으로 설정

cout << m.get() << ' '; // 500 출력

cout << m.isEven(); // true(정수로 1) 출력

}

**결과 및 결과분석**



자동으로 인라인 처리가 되어 생성자와 객체함수들이 작동합니다.

**소감**

자동 인라인 처리가 되면 실행시간이 빨라지는 장점도 있지만, 클래스 내부에 어떤 함수들이 구현되어 있는지 한눈에 확인하기 좋은 장점이 있는 것 같습니다.

**실습과제 10**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**문제분석**

클래스의 선언부와 구현부를 나눌 수 있는지 확인하는 문제입니다.

**프로그램 설계**

Class의 구조는 헤더파일에 선언하고, cpp파일에 자동 인라인 처리를 해줍니다.

**소스코드 및 주석**

**10 – 1)**

#include <iostream>

using namespace std;

class Add {

public:

void SetValue(int, int);

void calculate();

private:

int num1, num2;

};

class Sub {

public:

void SetValue(int, int);

void calculate();

private:

int num1, num2;

};

class Mul {

public:

void SetValue(int, int);

void calculate();

private:

int num1, num2;

};

class Div {

public:

void SetValue(int, int);

void calculate();

private:

int num1, num2;

};

// 같은 종류의 멤버변수와 메소드를 가지고 있는 객체들을 생성

void Add::SetValue(int a, int b) {

num1 = a;

num2 = b;

}

void Sub::SetValue(int a, int b) {

num1 = a;

num2 = b;

}

void Mul::SetValue(int a, int b) {

num1 = a;

num2 = b;

}

void Div::SetValue(int a, int b) {

num1 = a;

num2 = b;

}

// 매개변수의 값을 멤버 변수에 복사해주는 메소드 선언

void Add::calculate() {

cout << num1 + num2 << endl;

}

void Sub::calculate() {

cout << num1 - num2 << endl;

}

void Mul::calculate() {

cout << num1 \* num2 << endl;

}

void Div::calculate() {

if (num2 == 0) {

cout << "divided 0 error" << endl;

}

else cout << num1 / num2 << endl;

}

// 각각 다른값을 리턴하는 calcurate 메소드 구현, 0으로 나눌때는 에러 출력 후 0 리턴

int main() {

int num1, num2; // 입력받은 숫자가 저장될 변수

char op; // 계산 식이 입력될 변수

Add a;

Sub s;

Mul m;

Div d;

while (1) { // 무한반복

cout << "두 정수와 연산자를 입력하세요. >> ";

cin >> num1 >> num2 >> op;

switch (op) {

case '+':

a.SetValue(num1, num2);

a.calculate();

break;

case '-':

s.SetValue(num1, num2);

s.calculate();

break;

case '\*':

m.SetValue(num1, num2);

m.calculate();

break;

case '/':

d.SetValue(num1, num2);

d.calculate();

break;

default:

break;

}

}

}

**10 – 2) 헤더파일**

class Add {

public:

void SetValue(int, int);

void calculate();

private:

int num1, num2;

};

class Sub {

public:

void SetValue(int, int);

void calculate();

private:

int num1, num2;

};

class Mul {

public:

void SetValue(int, int);

void calculate();

private:

int num1, num2;

};

class Div {

public:

void SetValue(int, int);

void calculate();

private:

int num1, num2;

};

// class의 원형들을 선언

**10 – 2) cpp파일**

#include <iostream>

#include "실습과제10\_2\_Calculator.h" // class의 원형을 포함

using namespace std;

int main() {

int num1, num2; // 입력받은 숫자가 저장될 변수

char op; // 계산 식이 입력될 변수

Add a;

Sub s;

Mul m;

Div d;

while (1) { // 무한반복

cout << "두 정수와 연산자를 입력하세요. >> ";

cin >> num1 >> num2 >> op;

switch (op) {

case '+':

a.SetValue(num1, num2);

a.calculate();

break;

case '-':

s.SetValue(num1, num2);

s.calculate();

break;

case '\*':

m.SetValue(num1, num2);

m.calculate();

break;

case '/':

d.SetValue(num1, num2);

d.calculate();

break;

default:

break;

}

}

}

// 헤더파일에 선언된 메소드들을 구현

void Add::SetValue(int a, int b) {

num1 = a;

num2 = b;

}

void Sub::SetValue(int a, int b) {

num1 = a;

num2 = b;

}

void Mul::SetValue(int a, int b) {

num1 = a;

num2 = b;

}

void Div::SetValue(int a, int b) {

num1 = a;

num2 = b;

}

// 매개변수의 값을 멤버 변수에 복사해주는 메소드 선언

void Add::calculate() {

cout << num1 + num2 << endl;

}

void Sub::calculate() {

cout << num1 - num2 << endl;

}

void Mul::calculate() {

cout << num1 \* num2 << endl;

}

void Div::calculate() {

if (num2 == 0) {

cout << "divided 0 error" << endl;

}

else cout << num1 / num2 << endl;

}

// 각각 다른값을 리턴하는 calcurate 메소드 구현, 0으로 나눌때는 에러 출력 후 0 리턴

**결과 및 결과분석**

10 – 1

텍스트, 폰트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명

10 - 2

텍스트, 폰트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명

**실습과제 12**

텍스트, 전자제품, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 전자제품, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**문제분석**

클래스의 선언부와 구현부, main()함수를 분리할 수 있는지 확인하는 문제입니다.

**프로그램 설계**

클래스의 선언부를 헤더파일에 작성하고, ram.cpp파일에서 자동 인라인을 통해 구현한 뒤 main()에서 실행합니다.

헤더파일에 #ifndef를 사용해 여러 cpp파일이 h파일 포함할 수 있게 해 구현부와 main()함수도 연결 해 줍니다.

**프로그램 설계**

**헤더파일**

#ifndef RAM\_H

#define RAM\_H

#include <iostream>

using namespace std;

class Ram {

public:

Ram(); // mem 배열을 0으로 초기화하고 size 100\*1024로 초기화

~Ram(); // “메모리 제거됨” 문자열 출력

char read(int address); // address 주소의 메모리 바이트 리턴

void write(int address, char value); // address 주소에 한 바이트로 value 저장

private:

char mem[100 \* 1024]; // 100KB 메모리. 한 번지는 한 바이트이므로 char 타입사용

int size;

};

#endif

**ram.cpp**

#include "실습과제12\_ram.h"

Ram::Ram() {

for (int i = 0; i < 100 \* 1024; i++) {

mem[i] = 0;

}

size = 100 \* 1024;

} // mem 배열을 0으로 초기화하고 size 100\*1024로 초기화

Ram::~Ram() {

cout << "메모리 제거됨" << endl;

} // “메모리 제거됨” 문자열 출력

char Ram::read(int address) {

return mem[address];

} // address 주소의 메모리 바이트 리턴

void Ram::write(int address, char value) {

mem[address] = value;

} // address 주소에 한 바이트로 value 저장

**main.cpp**

#include "실습과제12\_ram.h"

int main() {

Ram ram;

ram.write(100, 20); // 100 번지에 20 저장

ram.write(101, 30); // 101 번지에 30 저장

char res = ram.read(100) + ram.read(101); // 20 + 30 = 50

ram.write(102, res); // 102 번지에 50 저장

cout << "102 번지의 값 = " << (int)ram.read(102) << endl; // 102 번지 값 출력

}

**결과 및 결과분석**

****

**소감**

선언부 구현부만 작성 되어있는 파일들이 있기 때문에 main()함수가 간단하게 작성됩니다.

객체확인이나 수정이 필요한 경우 그 부분의 파일만 확인하면 되기 때문에, 코드를 이해하기 쉬운 것 같습니다.

다른 cpp코드에서 클래스를 재사용 할 수 있다는 장점이 있는 것 같습니다.