REPORT



|  |  |
| --- | --- |
| 과 목 : | 객체지향프로그래밍 |
| 제출일자 : | 2023.05.16 |
| 담당교수 : | 황성호 |
| 학 과 : | 멀티디자인학과 |
| 학 번 : | 201522405 |
| 이 름 : | 최준하 |

1. (예제 7-1) 프렌드 함수 만들기.

#include <iostream>

using namespace std;

class \_rect;

bool equals(\_rect r, \_rect s);

class \_rect {

int width, height;

public:

\_rect(int width, int height) {

this->width = width; this->height = height;

}

friend bool equals(\_rect r, \_rect s);

};

bool equals(\_rect r, \_rect s) {

if (r.width == s.width && r.height == s.height) return true;

else

{

return false;

}

}

int main() {

\_rect a(3, 4), b(4, 5);

if (equals(a, b)) {

cout << "equal" << endl;

}

else

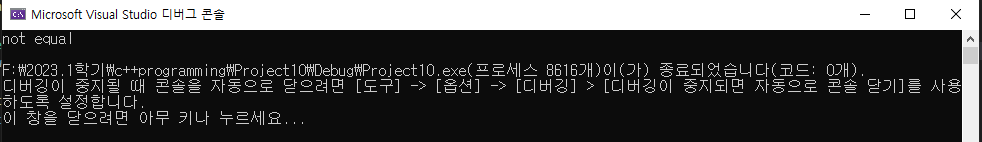
{

cout << "not equal" << endl;

}

}

🡪출력 결과



2. (예제 7-2) 다른 클래스의 멤버 함수를 프렌드 함수로 선언.

#include <iostream>

using namespace std;

class \_rect;

class \_rect\_manager {

public:

bool equals(\_rect r, \_rect s);

};

class \_rect {

int width, height;

public:

\_rect(int width, int height) { this->width = width; this->height = height; }

friend bool \_rect\_manager::equals(\_rect r, \_rect s);

};

bool \_rect\_manager::equals(\_rect r, \_rect s) {

if (r.width == s.width && r.height == s.height) {

return true;

}

else

{

return false;

}

}

int main() {

\_rect a(3, 4), b(3, 4);

\_rect\_manager man;

if (man.equals(a, b)) {

cout << "equal" << endl;

}

else

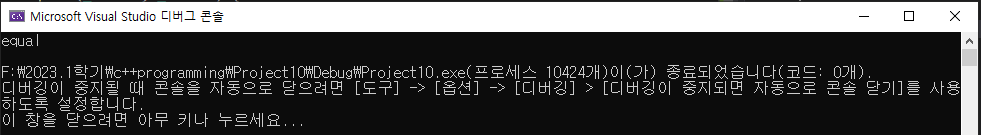
{

cout << "not equal" << endl;

}

}

🡪출력 결과



3. (예제 7-4) 두 개의 power 객체를 더하는 + 연산자 작성.

#include<iostream>

using namespace std;

class \_power {

int kick;

int punch;

public:

\_power(int kick = 0, int punch = 0) {

this->kick = kick; this->punch = punch;

}

void show();

\_power operator + (\_power op2);

};

void \_power::show() {

cout << "kick= " << kick << "," << "punch= " << punch << endl;

}

\_power \_power::operator+(\_power op2) {

\_power tmp;

tmp.kick = this->kick + op2.kick;

tmp.punch = this->punch + op2.punch;

return tmp;

}

int main() {

\_power a(3, 5), b(4, 6), c;

c = a + b;

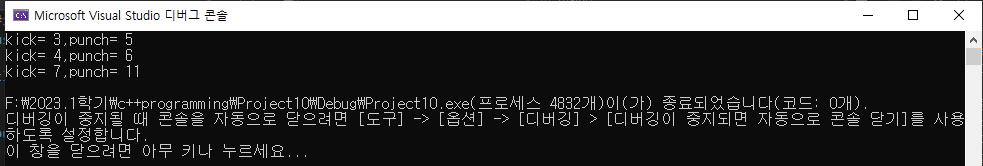
a.show();

b.show();

c.show();

}

🡪출력 결과



4. (예제 7-6) 두 개의 power 객체를 더하는 += 연산자 작성.

#include <iostream>

using namespace std;

class \_power {

int kick;

int punch;

public:

\_power(int kick = 0, int punch = 0) {

this->kick = kick; this->punch = punch;

}

void show();

\_power operator += (\_power op2);

};

void \_power::show() {

cout << "kick= " << kick << "," << "punch= " << punch << endl;

}

\_power \_power::operator+=(\_power op2) {

kick = kick + op2.kick;

punch = punch + op2.punch;

return \*this;

}

int main() {

\_power a(3, 5), b(4, 6), c;

a.show();

b.show();

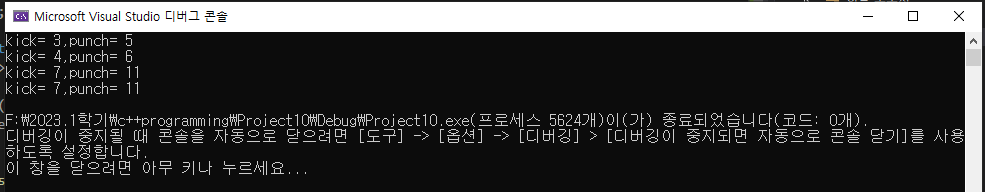
c = a += b;

a.show();

c.show();

}

🡪출력 결과



5. (예제 7-8) 전위 ++ 연산자 작성.

#include <iostream>

using namespace std;

class \_power {

int kick;

int punch;

public:

\_power(int kick = 0, int punch = 0) {

this->kick = kick; this->punch = punch;

}

void show();

\_power& operator ++ ();

};

void \_power::show() {

cout << "kick= " << kick << "," << "punch= " << punch << endl;

}

\_power& \_power::operator++() {

kick++;

punch++;

return \*this;

}

int main() {

\_power a(3, 5), b;

a.show();

b.show();

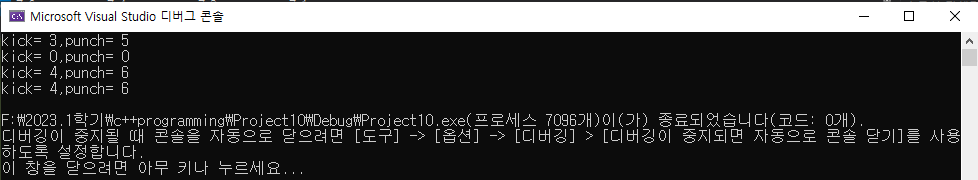
b = ++a;

a.show();

b.show();

}

🡪출력 결과



6. (예제 7-11) 2+a를 위한 + 연산자 함수를 프렌드로 작성.

#include <iostream>

using namespace std;

class \_power {

int kick;

int punch;

public:

\_power(int kick = 0, int punch = 0) {

this->kick = kick; this->punch = punch;

}

void show();

friend \_power operator+(int op1, \_power op2);

};

void \_power::show() {

cout << "kick= " << kick << "," << "punch= " << punch << endl;

}

\_power operator+(int op1, \_power op2) {

\_power tmp;

tmp.kick = op1 + op2.kick;

tmp.punch = op1 + op2.punch;

return tmp;

}

int main() {

\_power a(3, 5), b;

a.show();

b.show();

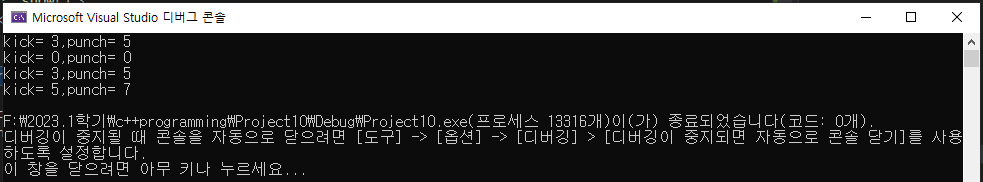
b = 2+a;

a.show();

b.show();

}

🡪출력 결과



7. (예제 7-14) 참조를 리턴하는 << 연산자 작성.

#include <iostream>

using namespace std;

class \_power {

int kick;

int punch;

public:

\_power(int kick = 0, int punch = 0) {

this->kick = kick; this->punch = punch;

}

void show();

\_power& operator << (int n);

};

void \_power::show() {

cout << "kick= " << kick << "," << "punch= " << punch << endl;

}

\_power& \_power::operator << (int n) {

kick += n;

punch += n;

return \*this;

}

int main() {

\_power a(1, 2);

a << 3 << 5 << 6;

a.show();

}

🡪출력 결과

