실습 10 - 11주차

학과 : 전자공학과 학번 : 2023104322 이름 : 현시온

- 과제는 pdf로 변환하여 제출(과제 문서 첫 줄에 학과/학번/이름 포함)
- 과제는 순서대로 작성하며, 문제와 설명을 모두 포함(형식이 맞지 않으면 감점)
- 프로그램을 작성하는 문제는 소스코드와 실행 결과를 모두 text로 붙여넣기(그림으로 포함하지 말 것)하고 코드 설명 및 결과에 대한 설명을 포함해야 함
- 문의 사항은 이메일(nize@khu.ac.kr) 또는 오픈 카톡방을 이용

1. 클래스 인스턴스를 아래와 같이 두가지(a와 p)로 선언하는 경우의 차이점에 대해서 설명하라.

```
class A { /* … */ };
A a;
A *p = new A;
```

설명:

전자는 정적(static) 메모리에 A 타입의 객체를 생성하는 반면, 후자는 동적 메모리에 A 타입의 객체를 생성하는 선언 방식이다. 전자는 선언이 끝나면 할당된 메모리가 자동으로 해제되지만, 후자는 delete 연산자를 통해 직접 할당된 메모리를 해제해주어야 한다.

2. 아래 코드에서 Vector2D는 2차원 벡터로 x, y 요소를 멤버로 가지는 클래스이다. 아래의 코드가 동작하도록 필요한 함수(메소드)를 추가하라.

```
#include <iostream>
class Vector2D {
   int x, y;
public:
   Vector2D(int x, int y) : x(x), y(y) {}
};
int main() {
   Vector2D v1(10, 2), v2(20, 5);
   std::cout << v1+v2 << std::endl; // 30, 7 출력
}
```

코드 및 설명:

```
#include <iostream>
class Vector2D { //클래스 Vector2D 선언 및 정의
        int x, y; //Vector2D의 private 멤버 변수 (정수형) x, y 선언
public:
        Vector2D(int x, int y) : x(x), y(y) {} //Vector2D의 멤버 변수 x, y를 생성자 파라미터 x, y로 초기화
        friend Vector2D operator+(Vector2D& a, Vector2D& b);
        friend std::ostream& operator<<(std::ostream& os, const Vector2D& V2D);
        //private 멤버 변수에 대한 접근 권한을 오버로드된 연산자 +와 <<가 얻기 위해 friend로 선언
};
```

```
Vector2D operator+(Vector2D& a, Vector2D& b) {
        //Vector2D의 객체 레퍼런스 a, b를 파라미터로 하여 Vector2D의 객체를 반환하는
오버로드된 연산자 +를 정의
        Vector2D result(0,0); //Vector2D의 객체 result를 멤버 변수 x, y를 0, 0으로 초기화하여
선언
        result.x = a.x + b.x;
        result.y = a.y + b.y;
        return result;
        //객체 result의 멤버 변수 x, y에 각각 객체 레퍼런스 a, b의 멤버 변수 x, y의 합을
할당하고 result를 반환
}
std::ostream& operator<<(std::ostream& os, const Vector2D& V2D) {
        //클래스 std의 멤버 ostream의 레퍼런스 os와 클래스 Vector2D의 const 객체 레퍼런스
V2D를 파라미터로 하여 클래스 std의 멤버 ostream의 레퍼런스를 반환하는 오버로드된 연산자 <<를
정의
        os << V2D.x << ',' << V2D.y;
        return os;
        //뒷 피연산자, 즉 V2D의 멤버 변수들을 출력하고, 앞 피연산자(std::cout)을 반환함
}
int main() {
        Vector2D v1(10, 2), v2(20, 5);
        std::cout << v1 + v2 << std::endl; // 30, 7 출력
}
```

3. 아래 코드가 동작하도록 클래스 A를 완성하라.

```
#include <iostream>
class A {
   int x;
public:
   A(int x) : x(x) {}
   // GetX, Max 함수 작성
};

int main() {
 const A a1(10);
 A a2(5), a3(3);
 std::cout << a1.GetX() << std::endl; // 10 출력
 std::cout << a2.GetX() << std::endl; // 5 출력

 A *p = a2.Max(&a3); // Max: a2.x와 a3.x를 비교하여
                              // 큰 값을 가지는 인스턴스 주소 반환
 std::cout << p->GetX() << std::endl; // 5출력
}
```

코드 및 설명:

```cpp
#include <iostream>
class A {
    int x;
public:
    A(int x) : x(x) {}
    // GetX, Max 함수 작성
    int GetX() const {
        return x;
    }
    //private 멤버 변수 x를 외부에서 사용하기 위해서 이의 복사본을 반환하는 const 메소드 GetX를 정의
    //메소드 GentX를 const로 정의하는 이유는 main에서 a1이 const 객체이기 때문이다
    A* Max(A* a) {
    //A의 객체의 포인터 a를 파라미터로 하여 A의 객체의 포인터, 즉 인스턴스의 주소를 반환하는 메소드 Max를 정의
        if (this->x > (*a).x) {
            return this;
        }
        else if (this->x < (*a).x) {
            return a;
        }
        //메소드 Max를 호출한 객체의 멤버 변수 x와 a의 대상의 멤버 변수 x 중 더 큰 쪽의 객체의 주소를 반환하도록 한다
    }
};
int main() {
    const A a1(10);
    A a2(5), a3(3);
    std::cout << a1.GetX() << std::endl; // 10 출력
    std::cout << a2.GetX() << std::endl; // 5 출력

    A* p = a2.Max(&a3); // Max: a2.x와 a3.x를 비교하여
    // 큰 값을 가지는 인스턴스 주소 반환
    std::cout << p->GetX() << std::endl; // 5출력
}
```

4. 클래스의 static 멤버의 특징과 사용법에 대해 설명하라.

   클래스의 모든 객체가 공유하는 멤버로, 변수의 경우 모든 객체가 동일한 값을 공유하며, 함수의 경우 객체 생성 없이도 '클래스 이름::함수 이름(인수);'를 통해 호출 가능하다. Stactic 멤버 변수는 클래스 내부에서 선언만 하고, 클래스 외부에서 '변수 타입 클래스 이름::변수 이름 = 값;'과 같이 초기화해야 한다. 한편 Static 멤버 함수는 클래스 내부에서 선언 및 정의하고, 위에서 설명한 것과 같이 객체 생성 없이도 호출 가능하다.

5. friend 함수와 friend 클래스에 대해 설명하라.

   Friend 함수와 클래스는 자신이 선언된, 즉 소속된 클래스의 private를 포함한 모든 멤버에 대해서 접근 권한을 얻는다. 선언의 경우 멤버에 대한 권한을 받을 클래스 내부에, 정의의 경우

그 클래스 외부에 작성하는데, 정의의 내용 및 파라미터로 권한을 받은 멤버를 사용할 수 있다고 볼 수 있다.

6. C++에서 class와 struct의 차이점에 대해서 설명하라.

C++에서 class와 struct은 모두 객체 지향 프로그래밍, 즉 우리가 흔히 클래스라고 부르는 기능을 수행할 수 있지만, struct의 경우 멤버의 접근 권한이 모두 public이지만, class의 경우 멤버의 접근 권한을 private와 public으로 나눌 수 있다.

7. 큰 범위의 0과 양의 정수를 저장하는 BigUnsigned 클래스를 아래와 같은 특징을 가지도록 정의하라. (테스트 코드도 작성해서 결과로 포함)

   A. 정수를 저장하는 멤버는 std::vector로 각 요소는 정수의 각 자릿수로 0-9까지의 정수를 가진다.

   B. 기본 생성자는 정수를 0으로 초기화한다.

   C. unsigned int 형을 파라메터로 하는 생성자를 가지며, 내부 정수를 인수로 초기화한다.

   D. std::string 형을 파라메터로 하는 생성자를 가지며, 내부 정수를 인수로 초기화하며 인수는 n번째 위치의 문자는 내부 정수의 n번째 자릿수이다.

   E. +연산자로 덧셈이 가능하며, std::cout과 <<을 이용하여 정수의 출력이 가능하다.

코드 및 설명:

```
#include <iostream>
#include <vector>
#include <cmath>

int digit(unsigned int num) {
    int count = 0;
    if (num == 0) {
        return 1;
    }
    while (num > 0) {
        num /= 10;
        count++;
    }
    return count;
}
//입력한 정수가 총 몇 자리인지 얻을 수 있는 함수이다.

int pick(unsigned int num, int pos) {
    for (int i = 0; i < pos; i++) {
        num /= 10;
    }
```

```
    num %= 10;
    return num;
}
//입력한 정수의 특정 자리의 수를 얻을 수 있는 함수이다.

class BigUnsigned {
    unsigned int l;
    std::vector<int> V;
    //입력받은 정수를 할당할 멤버 변수 l와 입력받은 정수를 저장할 멤버 벡터 V를 선언한다.
public:
    BigUnsigned() {
        l = 0;
        V.push_back(0);
    }
    //기본 생성자로는 마치 0을 입력받은 상태와 같이 l에 0을 할당하고 벡터 V는 단하나의 엘리먼트
0만을 가지도록 한다.
    BigUnsigned(unsigned int num) {
        l = num;
        V.resize(digit(num));
        for (int i = 0; i < digit(num); i++) {
            V[i] = pick(num, i);
        }
    }
    //unsigned int형 자료 num을 파라미터로 하는 생성자는 l에 num을 할당하고 num의 총 자리의
갯수만큼 V의 크기를 조정한 다음,
    //num의 각 자리가 엘리먼트의 번호와 대응하도록 V의 엘리먼트에 num의 각 자리의 수를
할당한다.
    BigUnsigned(const std::string& str) {
        unsigned int num = 0;
        for (int i = 0; i < str.size(); i++) {
            if (str[i] >= '0' && str[i] <= '9') {
                num = num * 10 + static_cast<int>(str[i] - '0');
            }
        }
        l = num;
        V.resize(digit(num));
        for (int i = 0; i < digit(num); i++) {
            V[i] = pick(num, i);
        }
    }
    //std::string형 자료 str을 파라미터로 하는 생성자는 unsigned int형 num을 0으로 선언 및
초기화하고, str의 n번째 문자가 num의 n번째 자리의 수가 되도록 num에 할당한 다음,
    //이후 num을 통해 unsigned int형 자료를 파라미터로 하여 정의한 생성자의 내용과 동일하다.
    std::vector<int> GetV() {
        return V;
    }
    unsigned int Getl() const {
        return l;
    }
    //V와 l의 값을 사용하기 위한 복사본을 반환하는 메소드다.
};
BigUnsigned operator+(const BigUnsigned& a, const BigUnsigned& b) {
    BigUnsigned result(a.Getl() + b.Getl());
    return result;
}
```

```
//오버로드된 연산자 +는 BigUnsigned의 객체간의 합이 가능하도록 하기 위해서 입력한 정수에
해당하는 I를 반환하는 GetI를 각각의 객체에서 호출하고,
// 그것을 합한 결과값을 인수로 하여 생성자를 호출한 결과를 반환하도록 구성하였다.
std::ostream& operator<<(std::ostream& os, const BigUnsigned& a) {
    os << a.GetI();
    return os;
}
//오버로드된 연산자 <<은 뒷 피연산자, 즉 BigUnsigned의 객체 a가 입력받은 정수를 출력하고, 앞
피연산자(std::cout)을 반환하도록 구성하였다.

int main() {
    std::cout << BigUnsigned() << std::endl;
    std::cout << BigUnsigned(2023104322) << std::endl;
    std::cout << BigUnsigned("2023104322") << std::endl;
    std::cout << BigUnsigned(2023104322) + BigUnsigned("2023104322") << std::endl;
    return 0;
}
```

출력 결과:

0

2023104322

2023104322

4046208644