[자료구조 과제 2]

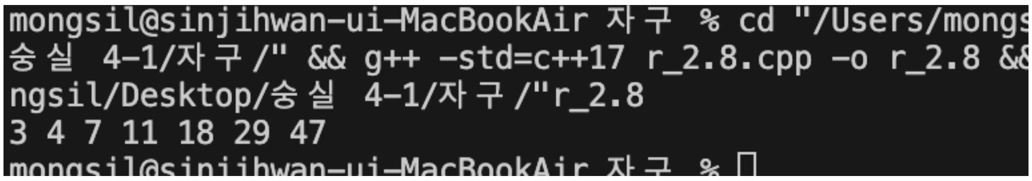
20202863 신지환

R-2.8 Give a short fragment of C++ code that uses the progression classes from Section 2.2.3 to find the 7th value of a Fibonacci progression that starts with 3 and 4 as its first two values

```
/*
R-2.8 Give a short fragment of C++ code
that uses the progression classes from Section 2.2.3
to find the 7th value of a Fibonacci progression that starts with 3 and 4 as its
first two values
*/


#include <iostream>

using namespace std;

class Progression {              // a generic progression
public:
    Progression(long f = 0)       // constructor
    : first(f), cur(f) { }
    virtual ~Progression() { };    // destructor
    void printProgression(int n);  // print the first n values
protected:
    virtual long firstValue();    // reset
    virtual long nextValue();     // advance
protected:
    long first;                 // first value
    long cur;                  // current value
};

class FibonacciProgression : public Progression {  // Fibonacci progression
public:
    FibonacciProgression(long f = 0, long s = 1);  // constructor
protected:
    virtual long firstValue();                  // reset
    virtual long nextValue();                   // advance
protected:
    long second;                             // second value
    long prev;                               // previous value
};

void Progression::printProgression(int n) {       // print n values
    cout << firstValue();                      // print the first
    for (int i = 2; i <= n; i++)                // print 2 through n
        cout << ' ' << nextValue();
    cout << endl;
```

```cpp
}

long Progression::firstValue() {                    // reset
   cur = first;
   return cur;
}

long Progression::nextValue() {                     // advance
   return ++cur;
}

FibonacciProgression::FibonacciProgression(long f, long s)
   : Progression(f), second(s), prev(second - first) { }

long FibonacciProgression::firstValue() {           // reset
   cur = first;
   prev = second - first;                           // create fictitious prev
   return cur;
}

long FibonacciProgression::nextValue() {            // advance
   long temp = prev;
   prev = cur;
   cur += temp;
   return cur;
}


/** Test program for the progression classes */
int main() {

   Progression* prog;

   prog = new FibonacciProgression(3, 4);
   prog->printProgression(7);

   return EXIT_SUCCESS; // successful execution
}
```

메인함수에서 피보나치 수열을 조건에 맞게 수정해 출력



```
mongsil@sinjihwan-ui-MacBookAir 자구 % cd "/Users/mongs
숭실 4-1/자구/" && g++ -std=c++17 r_2.8.cpp -o r_2.8 &&
ngsil/Desktop/숭실 4-1/자구/"r_2.8
3 4 7 11 18 29 47
mongsil@sinjihwan-ui-MacBookAir 자구 % 
```

R-2.10 Suppose we have a variable *p* that is declared to be a pointer to an object of type Progression using the classes of Section 2.2.3. Suppose further that *p* actually points to an instance of the class GeomProgression that was created with the default constructor. If we cast *p* to a pointer of type Progression and call *p*->firstValue(), what will be returned? Why?.

```
#include <iostream>

using namespace std;

class Progression {               // a generic progression
public:
   Progression(long f = 0)        // constructor
   : first(f), cur(f) { }
   virtual ~Progression() { };     // destructor
   void printProgression(int n);   // print the first n values
protected:
   virtual long firstValue();      // reset
   virtual long nextValue();       // advance
protected:
   long first;                   // first value
   long cur;                     // current value
};

class GeomProgression : public Progression {    // geometric progression
public:
   GeomProgression(long b = 2);              // constructor
protected:
   virtual long nextValue();                 // advance
protected:
   long base;                              // base value
};

void Progression::printProgression(int n) {        // print n values
   cout << firstValue();                        // print the first
   for (int i = 2; i <= n; i++)                  // print 2 through n
      cout << ' ' << nextValue();
   cout << endl;
}

long Progression::firstValue() {                 // reset
   cur = first;
   return cur;
}

long Progression::nextValue() {                  // advance
   return ++cur;
}


GeomProgression::GeomProgression(long b)          // constructor (Geometric
Progression)
   : Progression(1), base(b) { }

long GeomProgression::nextValue() {              // advance by multiplying
      cur *= base;
      return cur;
}




/** Test program for the progression classes */
int main() {

   Progression* p = new GeomProgression(); // GeomProgression 객체를 Progression*
포인터에 저장
   cout << p->firstValue() << endl; //접근 지정자 오류(protected)
   delete p;
   return 0;

   return EXIT_SUCCESS; // successful execution
}
```

firstValue()를 호출하는 부분에서 접근지정자가 protected로 되어있어 오류가 발생한다.

[{

"resource": "/Users/mongsil/Desktop/숭실 4-1/자구/r_2.10.cpp",

"owner": "C/C++: IntelliSense",

"code": "265",

"severity": 8,

"message": "함수 \"Progression::firstValue\" (선언됨 줄 36)에 액세스할 수 없습니다.",

"source": "C/C++",

"startLineNumber": 61,

"startColumn": 16,

"endLineNumber": 61,

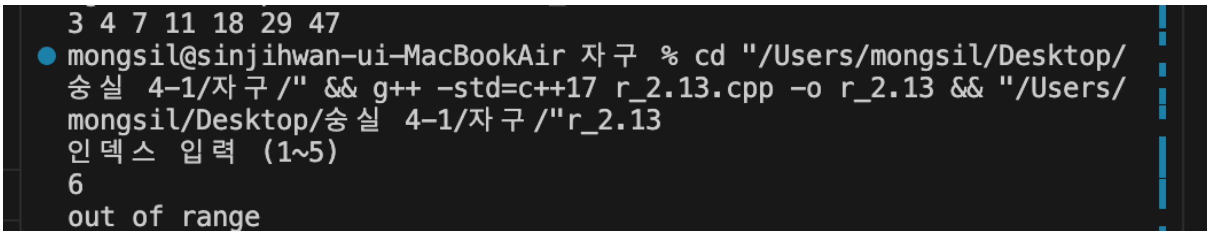
"endColumn": 26

}]

R-2.13 Give an example of a C++ code fragment that performs an array reference that is possibly out of bounds, and if it is out of bounds, the program catches that exception and prints an appropriate error message.

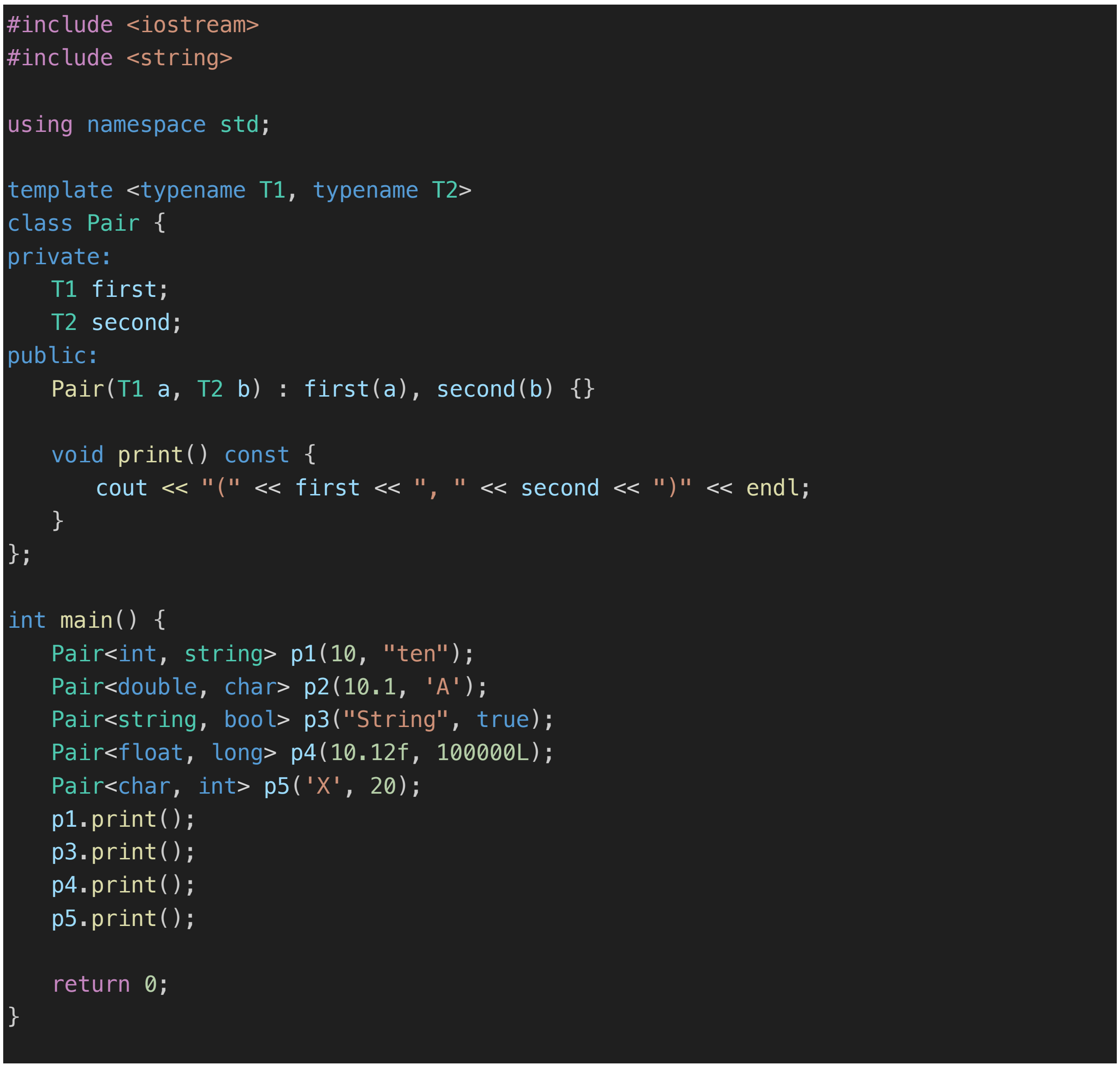
```cpp
#include <iostream>
#include <stdexcept>

using namespace std;

int main() {
    const int SIZE = 5;
    int arr[SIZE] = {1, 2, 3, 4, 5};

    int index;
    cout << "인덱스 입력 (1~5)" << endl;
    cin >> index;

    try {
        if (index < 0 || index >= SIZE) {
            throw out_of_range("out of range");
        }
        cout << "인덱스 " << index << "의 값: " << arr[index] << endl;
    } catch (const out_of_range& e) {
        cout << e.what() << endl;
    }

    return 0;
}
```

6 을 입력했을때 배열의 값을 벗어 났으므로 out of range 를 출력한다.



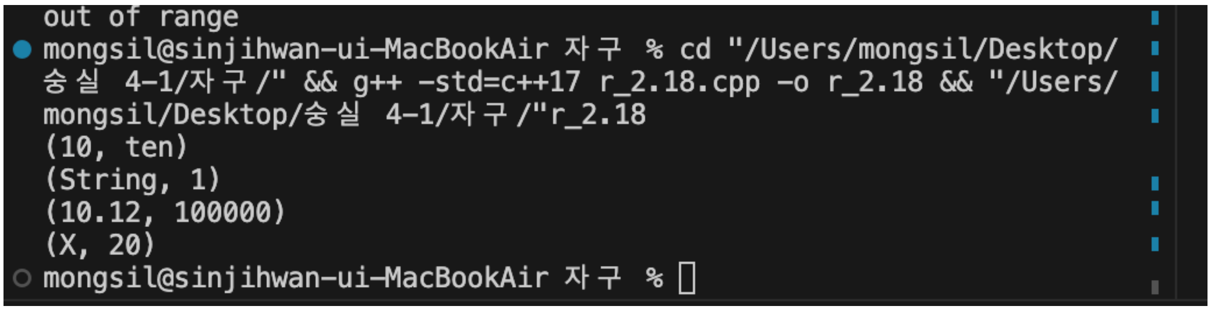
```
3 4 7 11 18 29 47
● mongsil@sinjihwan-ui-MacBookAir 자구 % cd "/Users/mongsil/Desktop/
숭실 4-1/자구/" && g++ -std=c++17 r_2.13.cpp -o r_2.13 && "/Users/
mongsil/Desktop/숭실 4-1/자구/"r_2.13
인덱스 입력 (1~5)
6
out of range
```

R-2.18 Write a short C++ program that creates a Pair class that can store two objects declared as generic types. Demonstrate this program by creating and printing Pair objects that contain five different kinds of pairs, such as <int,string> and <float,long>.



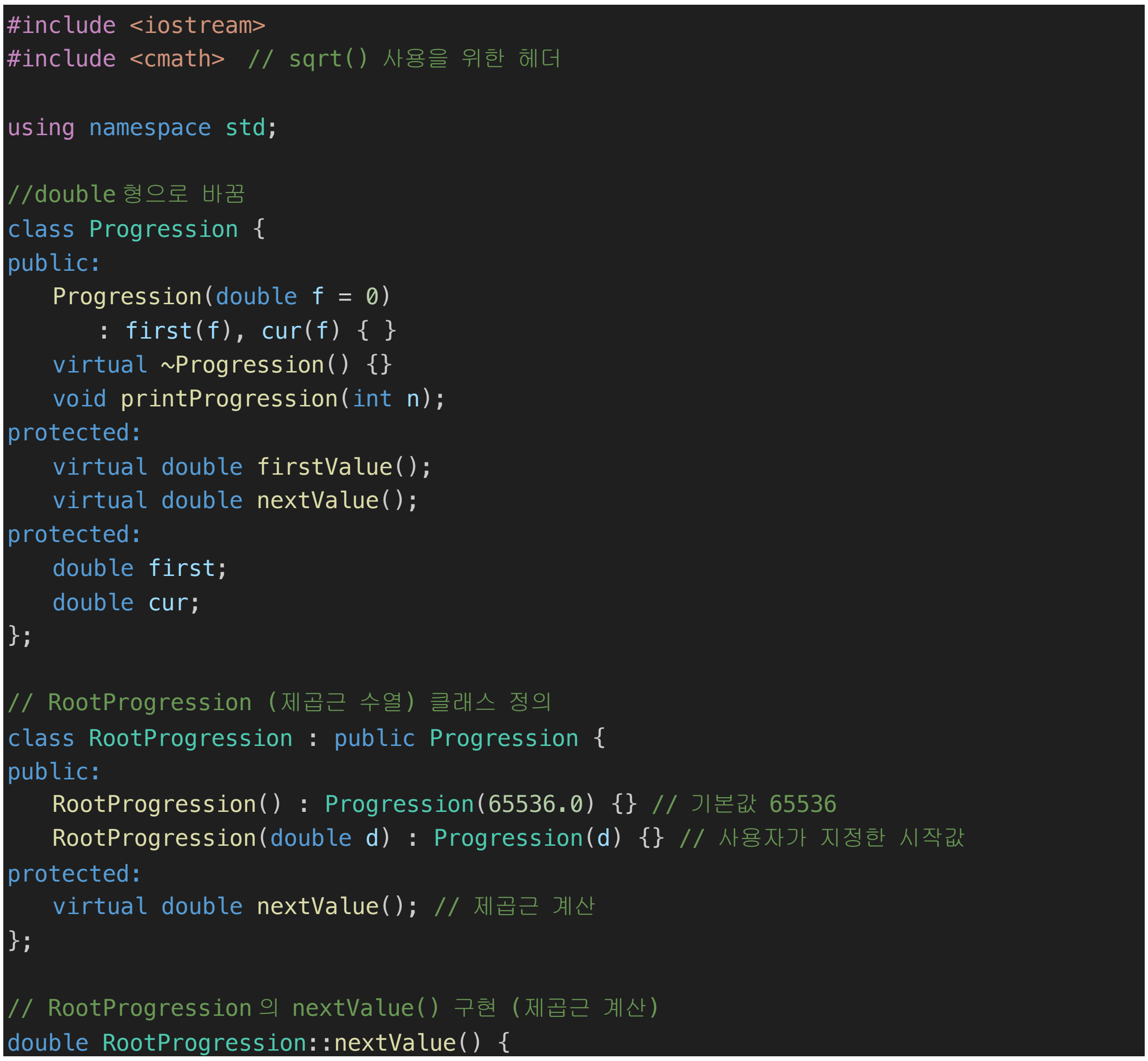
```cpp
#include <iostream>
#include <string>

using namespace std;

template <typename T1, typename T2>
class Pair {
private:
    T1 first;
    T2 second;
public:
    Pair(T1 a, T2 b) : first(a), second(b) {}

    void print() const {
        cout << "(" << first << ", " << second << ")" << endl;
    }
};

int main() {
    Pair<int, string> p1(10, "ten");
    Pair<double, char> p2(10.1, 'A');
    Pair<string, bool> p3("String", true);
    Pair<float, long> p4(10.12f, 100000L);
    Pair<char, int> p5('X', 20);
    p1.print();
    p3.print();
    p4.print();
    p5.print();

    return 0;
}
```

제네릭타입 두개를 사용해 여러 경우의 수를 테스트 출력해본 결과이다.



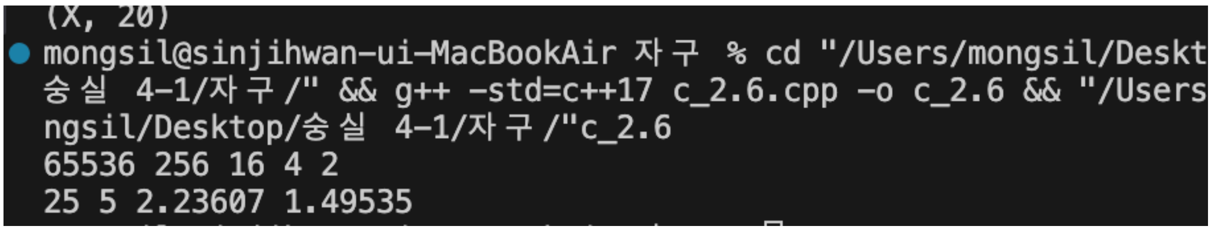
```
out of range
mongsil@sinjihwan-ui-MacBookAir 자구 % cd "/Users/mongsil/Desktop/
숭실 4-1/자구/" && g++ -std=c++17 r_2.18.cpp -o r_2.18 && "/Users/
mongsil/Desktop/숭실 4-1/자구/"r_2.18
(10, ten)
(String, 1)
(10.12, 100000)
(X, 20)
mongsil@sinjihwan-ui-MacBookAir 자구 %
```

C-2.6 Write a C++ class that is derived from the Progression class to produce a progression where each value is the square root of the previous value. (Note that you can no longer represent each value with an integer.) You should include a default constructor that starts with 65,536 as the first value and a parametric constructor that starts with a specified (double) number as the first value.



```cpp
#include <iostream>
#include <cmath>  // sqrt() 사용을 위한 헤더

using namespace std;

//double 형으로 바꿈
class Progression {
public:
    Progression(double f = 0)
        : first(f), cur(f) { }
    virtual ~Progression() {}
    void printProgression(int n);
protected:
    virtual double firstValue();
    virtual double nextValue();
protected:
    double first;
    double cur;
};

// RootProgression (제곱근 수열) 클래스 정의
class RootProgression : public Progression {
public:
    RootProgression() : Progression(65536.0) {} // 기본값 65536
    RootProgression(double d) : Progression(d) {} // 사용자가 지정한 시작값
protected:
    virtual double nextValue(); // 제곱근 계산
};

// RootProgression의 nextValue() 구현 (제곱근 계산)
double RootProgression::nextValue() {
    cur = sqrt(cur);
    return cur;
}

// Progression의 firstValue() 구현
double Progression::firstValue() {
    cur = first;
    return cur;
}


double Progression::nextValue() {
    return ++cur;
}


void Progression::printProgression(int n) {
    cout << firstValue();
    for (int i = 2; i <= n; i++)
        cout << " " << nextValue();
    cout << endl;
}

int main() {
    Progression* prog;
    prog = new RootProgression();
    prog->printProgression(5);
    delete prog;

    Progression* prog2;
    prog2 = new RootProgression(25);
    prog2->printProgression(4);
    delete prog2;

    return EXIT_SUCCESS;
}
```

기존에 있던 Progression 클래스에서 double형으로 받을 수 있게 수정했고 제곱근으로 줄어드는 수열을 만들어 테스트해보았다.



```
(X, 20)
● mongsil@sinjihwan-ui-MacBookAir 자구 % cd "/Users/mongsil/Deskt
숭실 4-1/자구/" && g++ -std=c++17 c_2.6.cpp -o c_2.6 && "/Users
ngsil/Desktop/숭실 4-1/자구/"c_2.6
65536 256 16 4 2
25 5 2.23607 1.49535
```