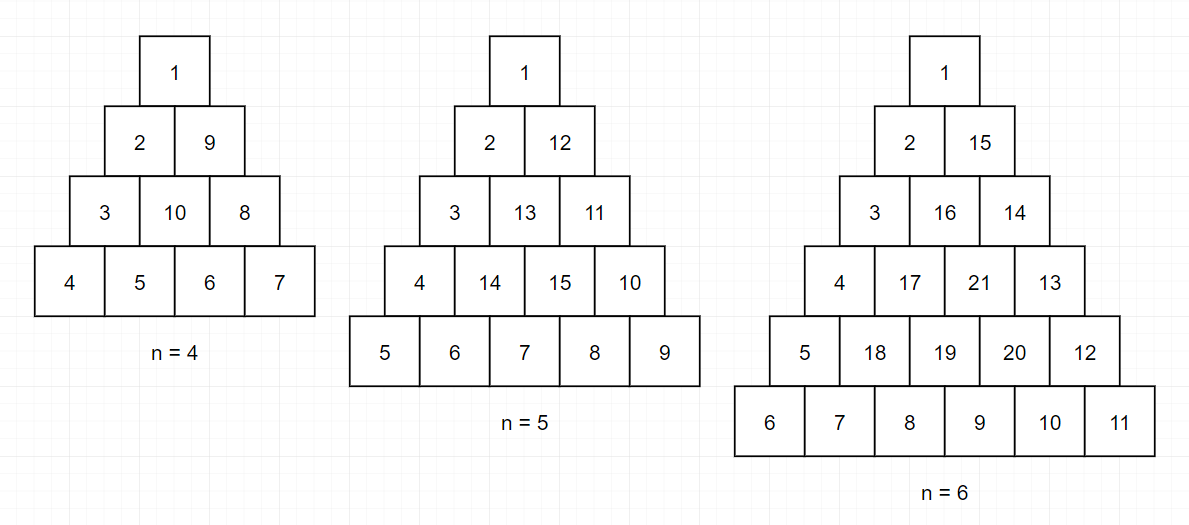
21/01/07

1. 삼각 달팽이

**문제 설명**

정수 n이 매개변수로 주어집니다. 다음 그림과 같이 밑변의 길이와 높이가 n인 삼각형에서 맨 위 꼭짓점부터 반시계 방향으로 달팽이 채우기를 진행한 후, 첫 행부터 마지막 행까지 모두 순서대로 합친 새로운 배열을 return 하도록 solution 함수를 완성해주세요.



제한사항

* n은 1 이상 1,000 이하입니다.

입출력 예

| **n** | **result** |
| --- | --- |
| 4 | [1,2,9,3,10,8,4,5,6,7] |
| 5 | [1,2,12,3,13,11,4,14,15,10,5,6,7,8,9] |
| 6 | [1,2,15,3,16,14,4,17,21,13,5,18,19,20,12,6,7,8,9,10,11 |

* 내 답

#include <string>

#include <vector>

#include <iostream>

using namespace std;

void snail(int n, int a, int b, int t,vector<vector<int>> &v, int c) {

int i,j;

vector<int> answer;

for(i=a;i<a+n;i++)

v[i][b] = t++;

i--;

for(j=b+1;j<b+n;j++)

v[i][j] = t++;

j--;

for(int s=0;s<n-2;s++)

v[--i][--j] = t++;

if(t==(c\*(c+1))/2 + 1) {

return;

}

else

snail(n-3,i+1,j,t,v,c);

}

vector<int> solution(int n) {

vector<int> answer;

vector<vector<int>> v(n,vector<int> (n,0));

snail(n,0,0,1,v,n);

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

for(int j=0;j<i+1;j++)

answer.push\_back(v[i][j]);

return answer;

}

구현하고 변수 초기화 문제로, 참조를 시키는 문제에 대해 고민하다가. 벡터만 주소로 참조받아서 해결하였다.

* 관련 내용

### C++로 프로그래밍할 때 인자 또는 리턴 값으로 std::vector 등 STL 데이터 타입 처리하는 좋은 방법

C++ 프로그래밍할 때 std::vector 등 STL 데이터 타입을 호출하는 함수로 넘겨주거나 받아야 할 경우가 있다. 이런 경우 메모리 문제 또는 속도 문제 때문에 고민을 많이 할 것 같은데, C++11 이상을 지원하는 컴파일러를 사용하는 분이라면 이 글에서 도움을 받아 가기 바란다.

#### 함수 인자로 넘길 때

이 경우는 사용이 명확하다. Call by Reference으로 데이터를 넘기면 된다.

Call By Reference로 인자로 넘기는 방법은 pointer 를 사용해도 되지만, reference을 사용하는 것이 좋다.

void foo1(std::vector& data)

{

  ...

}

void foo2(std::vectorconst& data)

{

  ...

}

void foo3(std::vector data)

{

}

int main(int argc, char\* argv[])

{

  std::vector my\_data {10, 20, 30};

  foo1(my\_data);

  foo2(my\_data);

}

위 코드에서 foo1()과 foo2()는 Call by Reference로 값을 넘기는 반면에 foo3()는 Call by Value로 main()에서 my\_data를 넘기게 된다.

foo1()의 data는 main()의 my\_data를 참조(reference)하고 있어서 foo1()에서의 데이터 변경은 main()의 my\_data에 영향을 주게 된다.

그리고 foo2()는 main()의 my\_data를 참조는 하지만 수정을 할 수가 없는 것이라서 단순히 값만 참조할 수 있게 된다.

반면 foo3()에서의 data는 main()의 my\_data를 넘기는 순간에 값을 복사하기 때문에 메모리가 증가하게 된다. 또한 foo3()에서 data를 수정하더라도 my\_data에 영향을 주지 않는다.

#### 함수에서 데이터로 받기

이 과정에서 개발자들은 많은 고민을 하게 된다. 다음 코드를 보자.

std::vector foo()

{

  std::vector data { 10, 20, 30};

  return data;

}

int main(int argc, char\* argv[])

{

  std::vector my\_data = foo();

}

이런 경우에 과연 my\_data에 값을 제대로 받을 수 있을까 생각할 수 있다.

결론은! 컴파일러가 데이터를 Call by Reference  형태로 foo()의 data의 값들이 my\_data에 이동이 되어 효율적으로 전달이 된다.

그리고 C++11 이전 버전과 호환되는 방법은 다음과 같다.

void foo(std::vector& data)

{

  data.push\_back(10);

  data.push\_back(20);

  data.push\_back(30);

}

int main(int argc, char\* argv[])

{

  std::vector my\_data;

 foo(my\_data);

}

1. 모의고사

###### 문제 설명

수포자는 수학을 포기한 사람의 준말입니다. 수포자 삼인방은 모의고사에 수학 문제를 전부 찍으려 합니다. 수포자는 1번 문제부터 마지막 문제까지 다음과 같이 찍습니다.

1번 수포자가 찍는 방식: 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, ...  
2번 수포자가 찍는 방식: 2, 1, 2, 3, 2, 4, 2, 5, 2, 1, 2, 3, 2, 4, 2, 5, ...  
3번 수포자가 찍는 방식: 3, 3, 1, 1, 2, 2, 4, 4, 5, 5, 3, 3, 1, 1, 2, 2, 4, 4, 5, 5, ...

1번 문제부터 마지막 문제까지의 정답이 순서대로 들은 배열 answers가 주어졌을 때, 가장 많은 문제를 맞힌 사람이 누구인지 배열에 담아 return 하도록 solution 함수를 작성해주세요.

##### 제한 조건

* 시험은 최대 10,000 문제로 구성되어있습니다.
* 문제의 정답은 1, 2, 3, 4, 5중 하나입니다.
* 가장 높은 점수를 받은 사람이 여럿일 경우, return하는 값을 오름차순 정렬해주세요.

##### 입출력 예

| **answers** | **return** |
| --- | --- |
| [1,2,3,4,5] | [1] |
| [1,3,2,4,2] | [1,2,3] |

* 내 답

#include <string>

#include <vector>

#include <iostream>

using namespace std;

vector<int> solution(vector<int> answers) {

vector<int> answer;

vector<int> a\_p{1,2,3,4,5};

vector<int> b\_p{2,1,2,3,2,4,2,5};

vector<int> c\_p{3,3,1,1,2,2,4,4,5,5};

int a=0,b=0,c=0;

for(int i=0;i<answers.size();i++) {

if(answers[i] == a\_p[i%5])

a++;

if(answers[i] == b\_p[i%8])

b++;

if(answers[i] == c\_p[i%10])

c++;

}

int max = a;

answer.push\_back(1);

if(b > a) {

max = b;

answer.pop\_back();

answer.push\_back(2);

}

else if(b == a)

answer.push\_back(2);

if(c > max) {

while(!answer.empty())

answer.pop\_back();

answer.push\_back(3);

}

else if(c == max)

answer.push\_back(3);

return answer;

}