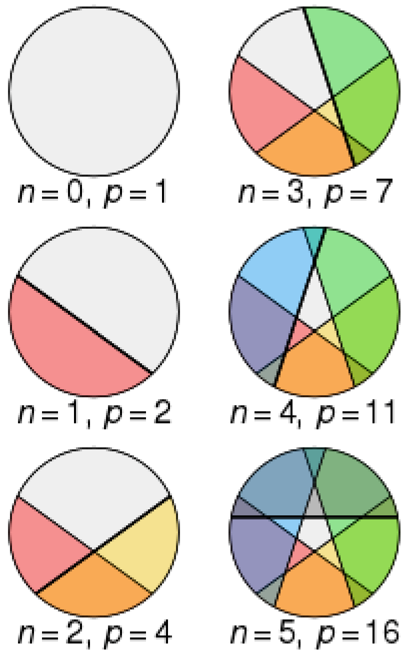
Midterm Programming Project: Central Polygonal Numbers
Topics: Classes, Array Allocation and Deallocation, File Output
CPSC-298-6 Programming in C++
jbonang@chapman.edu

Introduction

Can you make just three cuts across a pizza and produce seven pieces? Yes.

The figures below show you how. "n" is the number of cuts and "p" is the number of pieces produced.



It turns out that the maximum number of pieces, p, that can be created with a given number of cuts, n, where n ≥ 0, is given by the formula:

$$p = \frac{n^2 + n + 2}{2}$$

The formula yields a sequence of integer numbers. Starting with n = 0, the sequence is:

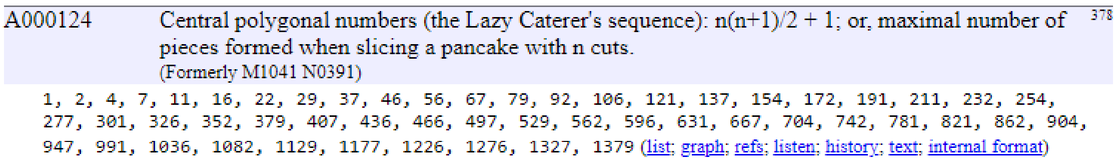
1, 2, 4, 7, 11, 16, 22, 29, 29, 37, 46, 56, 67, 79, ...

These are known as Central Polygonal Numbers, though the sequence is better known as the Lazy Caterer's Sequence. (If someone's lazy, and a caterer, it comes in handy.)

Mathematician Neil Sloan started collecting integer sequences such as this in 1965 when a graduate student. He originally published them in book form as *The Handbook of*

Integer Sequences, in 1973 - a page-turner if you're in to combinatorics. He later moved the handbook online as the *On-Line Encyclopedia of Integer Sequences*. As of July, 2020, this database of integer sequence has 336,000 sequences. The Lazy Caterer's Sequence is OEIS Sequence A000124 (https://oeis.org/A000124).

Here is the top of the entry for A000124:



A000124 Central polygonal numbers (the Lazy Caterer's sequence): n(n+1)/2 + 1; or, maximal number of pieces formed when slicing a pancake with n cuts. 378
(Formerly M1041 N0391)

```
1, 2, 4, 7, 11, 16, 22, 29, 37, 46, 56, 67, 79, 92, 106, 121, 137, 154, 172, 191, 211, 232, 254,
277, 301, 326, 352, 379, 407, 436, 466, 497, 529, 562, 596, 631, 667, 704, 742, 781, 821, 862, 904,
947, 991, 1036, 1082, 1129, 1177, 1226, 1276, 1327, 1379 (list; graph; refs; listen; history; text; internal format)
```

Notice that the entry includes the first few entries in the sequence: 1, 2, 4, 7 and so on.

Suppose you wanted to build up such a database (and who wouldn't), how would you go about it.

Well, you might want to save the first few entries of the sequence to a file after generating them with a C++ program.

The Assignment

For this project, you'll create a C++ class that computes the first N entries in the Lazy Caterer's Sequence, OEIS Sequence A000124. The sequence will be stored in an allocated array of integers within the class and then saved to a file.

Your C++ class should be similar to the following:

```
class CentralPolygonalNumbers
{
public:
    CentralPolygonalNumbers();
    CentralPolygonalNumbers(int nMax);
    ~CentralPolygonalNumbers();
    void display();
    bool save(std::string strFilename);

private:
    int m_nMax;
    int* mp_iNumbers;

};
```

The class must include a private member variable that is a pointer to an integer. This variable will eventually point to an allocated array of integers whose size is large enough to hold a specified number of the first entries in the sequence. (mp_iNumbers is used for this in the example above.)

The largest value of n to be used in computing the first elements of the sequence must be stored in a private integer member variable (m_nMax is used for this in the example).

The Class Constructors

Your class must include a default constructor and an overloaded constructor. The overloaded constructor accepts as an argument the highest (or maximum) number of n you wish to go up to in when computing the first few entries of the sequence. (It's shown as parameter nMax in the example above and stored as the private member m_nMax.) For example, if nMax is 3, then the sequence will contain 1, 2, 4, 7. (Note that though nMax is 3, there are four entries. This is because the sequence starts with n == 0. n, the number of cuts, takes on the values 0, 1, 2, 3 and, p, the number of pieces are 1, 2, 4, 7.)

The default constructor must assume that the value for m_nMax is 0. That is, the sequence contains only one entry, 1 (the value of p when n == 0).

Within the constructor, you'll need to allocate the array used to store the sequence. (Remember that the array will be one larger than the value nMax.) Use the new operator when allocating the array of integers. The statement to allocate the array of integers will appear similar to the following; you'll need to fill in the blank where the red question mark appears.

```
mp_iNumbers = new int[    ?    ];
```

The constructor should also use a for loop to calculate all the values in the sequence, the number of pieces, p. The loop must iterate over the values of n starting at 0 and up to and include m_nMax. It must store the computed values of p in the array of integers (pointed to by mp_iNumbers in the example code).

The Class Destructor

Your class must also contain a class destructor. The destructor must deallocate the memory allocated for the sequence (pointed to by mp_iNumbers in the example).

Remember that mp_iNumbers is a pointer to an array; how should you deallocate it,

`delete`
or
`delete[]`
?

At the end of the destructor, output a message to standard output using std::cout:

```
std::cout << "~CentralPolygonalNumbers called" << std::endl;
```

This way, you'll be able to tell that the destructor was actually called (behind the scenes).

Other Public Member Functions

The class must expose public member functions that display the sequence to standard output and save the sequence to a file.

To display the sequence to standard output, iterate over the array of integers containing the sequence and print out each value.

To save the sequence to a file, use an output file steam object:

```
std::ofstream ofsNumbers;
```

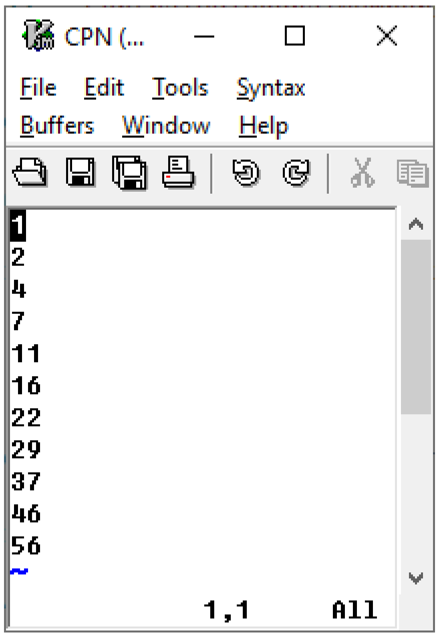
Use a for loop to iterate over the integer array containing the sequence. Use the insertion operator to write each number (the number of pieces) to the output file stream.

Remember to close the file after all entries in the sequence have been written.

```
ofsNumbers.close();
```

The file name must be passed as an argument. The file will appear in the program's current working directory.

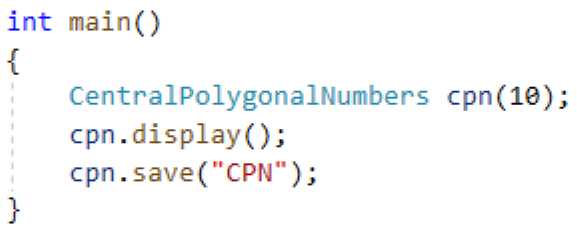
Here's an example of what the file looks like (shown in the vim editor) for sequence entries up to 56.



```
1
2
4
7
11
16
22
29
37
46
56
~
              1,1         All
```

The main Program

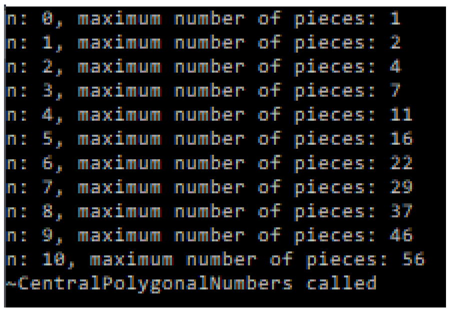
For the main program, supply the value 10 to the overloaded class constructor, then call your function to display the sequence to standard output, then call your function to save the sequence to a file. Make sure to supply a filename.



```
int main()
{
    CentralPolygonalNumbers cpn(10);
    cpn.display();
    cpn.save("CPN");
}
```

The file will appear in the directory in which your program is running.

The output of the program displayed on the screen will appear similar to the following.



```
n: 0, maximum number of pieces: 1
n: 1, maximum number of pieces: 2
n: 2, maximum number of pieces: 4
n: 3, maximum number of pieces: 7
n: 4, maximum number of pieces: 11
n: 5, maximum number of pieces: 16
n: 6, maximum number of pieces: 22
n: 7, maximum number of pieces: 29
n: 8, maximum number of pieces: 37
n: 9, maximum number of pieces: 46
n: 10, maximum number of pieces: 56
~CentralPolygonalNumbers called
```

Post your source code along with the saved file. Include a screen capture of the console output. Submit the project as you would any other project.

The next time you order a pizza from Domino's, tell them you want seven pieces but that they can only make three cuts. (If the chef is a computer science student working part-time, they just might do it. Otherwise - free pizza!   Maybe.)