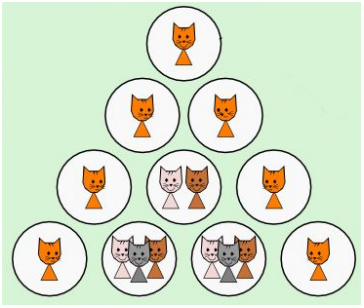
Trójkot Pascala



#kombinatoryka
#dwumian_Newtona

Trójkąt Pascala* to symetryczna trójkątna tablica utworzona z liczb naturalnych. Na brzegach (ramionach) znajdują się jedynki, a każda pozostała liczba jest sumą dwóch stojących nad nią. Oto pierwsze siedem wierszy tej tablicy (początkowy wiersz ma numer 0):

```
0                1
1              1   1
2            1   2   1
3          1   3   3   1
4        1   4   6   4   1
5      1   5  10  10  5   1
6    1   6  15  20  15  6   1
              ...
```

Liczby znajdujące się w trójkącie Pascala pojawiają się w wielu zagadnieniach matematycznych, a zwłaszcza w kombinatoryce. Znajdziemy je choćby w rozwinięciu dwumianu Newtona. Czy pamiętacie wzory skróconego mnożenia na kwadrat czy sześcian sumy dwóch składników? Współczynniki przy kolejnych wyrazach to właśnie elementy odpowiednich wierszy trójkąta Pascala, na przykład wiersza drugiego:

$$(a+b)^2 = \mathbf{1}\cdot a^2 + \mathbf{2}\cdot ab + \mathbf{1}\cdot b^2,$$

czy wiersza trzeciego:

$$(a+b)^3 = \mathbf{1}\cdot a^3 + \mathbf{3}\cdot a^2b + \mathbf{3}\cdot ab^2 + \mathbf{1}\cdot b^3.$$

Napiszemy teraz procedurę, która obliczy i wypisze kolejne wiersze trójkąta od zerowego po n-ty, gdzie n będzie argumentem procedury.

Zawartość wiersza, który właśnie obliczamy, trzeba przechować w jakiejś tablicopodobnej strukturze danych. Ponieważ każdy kolejny wiersz jest dłuższy od poprzedniego – czyli będzie rosnąć w trakcie biegu programu – zatem wybierzemy niedawno poznany typ danych: `vector<int>`. Na początku mamy taką oto sytuację:

```
int row{ };
vector<int> T(1, 1);
```

*Było dwóch uczonych o tym nazwisku: ojciec (Étienne Pascal) oraz syn (Blaise Pascal). Pierwszemu z nich zawdzięczamy opisanie i zbadanie kardioidy: krzywej zwanej też ślimakiem Pascala. Żył na przełomie XVI i XVII wieku. Syn wniósł ogromny wkład w rozwój fizyki i matematyki, doceniany do dzisiaj.

Zmienna *row* (wymawiaj: *roł*, od angielskiego „*wiersz*") oznacza numer bieżącego wiersza. Wektor T jest na razie jednoelementowy i zawiera tylko wartość 1 (ta druga jedynka w deklaracji oznacza właśnie nadaną wartość).

Gotowy wiersz trzeba wypisać, co możemy uczynić taką pętlą:

```
for(int x : T)
  cout << x << ' ';
cout << '\n';
```

Użycie znaku `\n` jest konieczne, aby kolejny wiersz został wypisany w następnej linijce na ekranie.

Teraz obliczamy następny wiersz: powiększamy zmienną *row* i dodajemy na końcu wektora T nowy element o wartości 1:

```
row++;
T.push_back(1);
```

Musimy wyliczyć (w pętli) kolejne elementy nowego wiersza. Ale uwaga: nie powinniśmy robić tego od lewej do prawej, będziemy bowiem zamazywać wartości potrzebne do obliczenia kolejnych elementów (dlaczego?). Prawidłowa kolejność to od prawej do lewej, a dokładniej od elementu przedostatniego (o numerze $row - 1$) aż do elementu o numerze 1.[†] W wierszu o numerze *row* ilość elementów wynosi $row + 1$, przy czym na jego obydwu końcach stoją liczby 1, których nie trzeba zmieniać. Każdy nowy element będzie sumą dotychczasowego elementu o tym numerze i jego sąsiada z lewej strony:

```
for(int i{row-1}; i >= 1; i--)
  T[i] = T[i] + T[i - 1];
```

albo, jak kto woli:

```
for(int i{row-1}; i >= 1; i--)
  T[i] += T[i - 1];
```

Teraz musimy tylko dodać pętlę po numerach wierszy i nasza procedura jest gotowa:

```
void pascal(int n)
{
  int row{ };
  vector<int> T(1, 1);
  while(row <= n)
  {
    // wypisywanie bieżącego wiersza
    for(int x : T)
      cout << x << ' ';
```

[†]Pamiętamy, że elementy wektora są numerowane od 0.

```
    cout << '\n';
    // obliczanie następnego wiersza
    row++;
    T.push_back(1);
    for(int i{row-1}; i >= 1; i--)
      T[i] += T[i - 1];
  }
}
```

Jeśli dopiszemy stosowny `main()` oraz dodamy odpowiedni nagłówek, wtedy na przykład dla $n = 6$ dostaniemy wydruk:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
```

No, nie wygląda to tak ładnie i symetrycznie, jak obrazek na początku niniejszego podrozdziału, ale zawiera wszystko to, co trzeba.

Liczby tworzące trójkąt Pascala wyglądają na niezbyt wielkie, ale proszę sprawdzić, co się dzieje, gdy wybierzemy większą wartość n, na przykład powyżej 30. Po pierwsze, linijki tekstu przestaną mieścić się na ekranie i zaczną się zawijać. To jeszcze detal, ale wśród wypisywanych liczb zaczną się pojawiać liczby ujemne. Skąd to? Otóż elementy trójkąta tak naprawdę wyrażają się przez symbole Newtona zawierające silnię, którą cechuje wykładnicze tempo wzrostu. Szybko zatem dochodzi do przekroczenia zakresu typu `int` i perturbacji w obliczeniach, choćby to było zwykłe dodawanie. Możemy zamienić typ `int` na `long long`, ale to wiele nie da, tylko opóźnimy ten arytmetyczny armagedon.

Proponujemy jednak inną zabawę: proszę zwrócić uwagę, że w trójkącie występują zarówno liczby parzyste, jak i nieparzyste. Skupmy się na samej parzystości liczb i tylko taką informację przechowujmy w wektorze T, wtedy nie będzie problemów z przepełnieniem. Przechowywać będziemy tylko resztę z dzielenia danej liczby przez 2 (czyli 0 lub 1). W tym celu musimy zmodyfikować sposób obliczania elementów nowego wiersza. Powinno być tak:

```
for(int i{row-1}; i >= 1; i--)
  T[i] = (T[i] + T[i - 1]) % 2;
```

Zmienimy także sposób prezentacji trójkąta na ekranie: zamiast liczby nieparzystej wypiszemy znak gwiazdki `*`, a zamiast liczby parzystej – spację:

```
for(int x : T)
  if(x)
    cout << '*';
  else
    cout << ' ';
cout << '\n';
```

Pamiętamy, że w razie czego, to 0 oznacza fałsz, a 1[‡] oznacza prawdę.

Jeśli teraz uruchomimy nasz program i wybierzemy na przykład $n = 31$, wówczas zobaczymy coś takiego:

```
*
**
* *
****
*   *
**  **
* * * *
********
*       *
**      **
* *     * *
****    ****
*   *   *   *
**  **  **  **
* * * * * * * *
****************
*               *
**              **
* *             * *
****            ****
*   *           *   *
**  **          **  **
* * * *         * * * *
********        ********
*       *       *       *
**      **      **      **
* *     * *     * *     * *
****    ****    ****    ****
*   *   *   *   *   *   *   *
**  **  **  **  **  **  **  **
* * * * * * * * * * * * * * * *
********************************
```

To trójkąt Sierpińskiego,[§] jeden z przykładów fraktali, trochę tylko przekrzywiony w lewo. Trójkąt Pascala kryje w sobie wiele matematycznych niespodzianek...

Czytelnikowi pozostawiamy domyślenie się, dlaczego wybrano właśnie $n = 31$ (inne liczby o tej własności, to: 7, 15, 63, ...).

Uważny Czytelnik na pewno spostrzeże, że w procedurze `pascal()` niepotrzebnie obliczamy wiersz o numerze $n+1$, ale w końcu to tylko kilkadziesiąt liczb, nie bądźmy zbyt drobiazgowi...

[‡]Dokładniej: każda liczba różna od zera.

[§]Tak naprawdę, to jedna z jego początkowych iteracji.