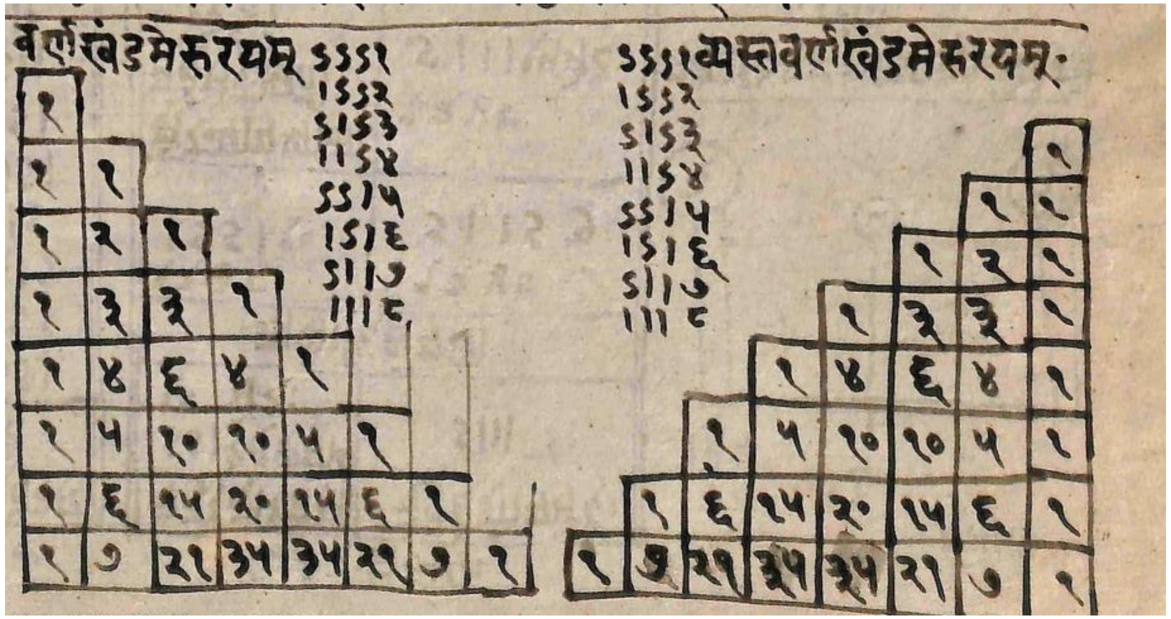
Meru Prastāra (775 CE, Kāshmir)

C(0,0)
C(1,0) C(1,1)
C(2,0) C(2,1) C(2,2)
C(3,0) C(3,1) C(3,2) C(3,3)
C(4,0) C(4,1) C(4,2) C(4,3) C(4,4)
C(5,0) C(5,1) C(5,2) C(5,3) C(5,4) C(5,5)



Each row k (from 0), gives C(k,0), C(k,1),..,C(k,l),…,C(k,k)
Number of ways l “long” symbols can be selected in k

```
            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
1   7  21  35  35  21   7   1
1   8  28  56  70  56  28   8   1
```

Figure 0.12: The Meru-Prastāra

Meru mod 2

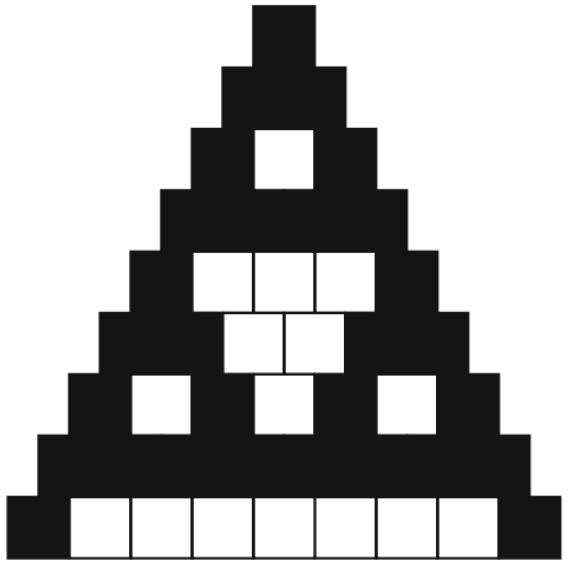


Figure 0.15: The Meru-Prastāra mod 2

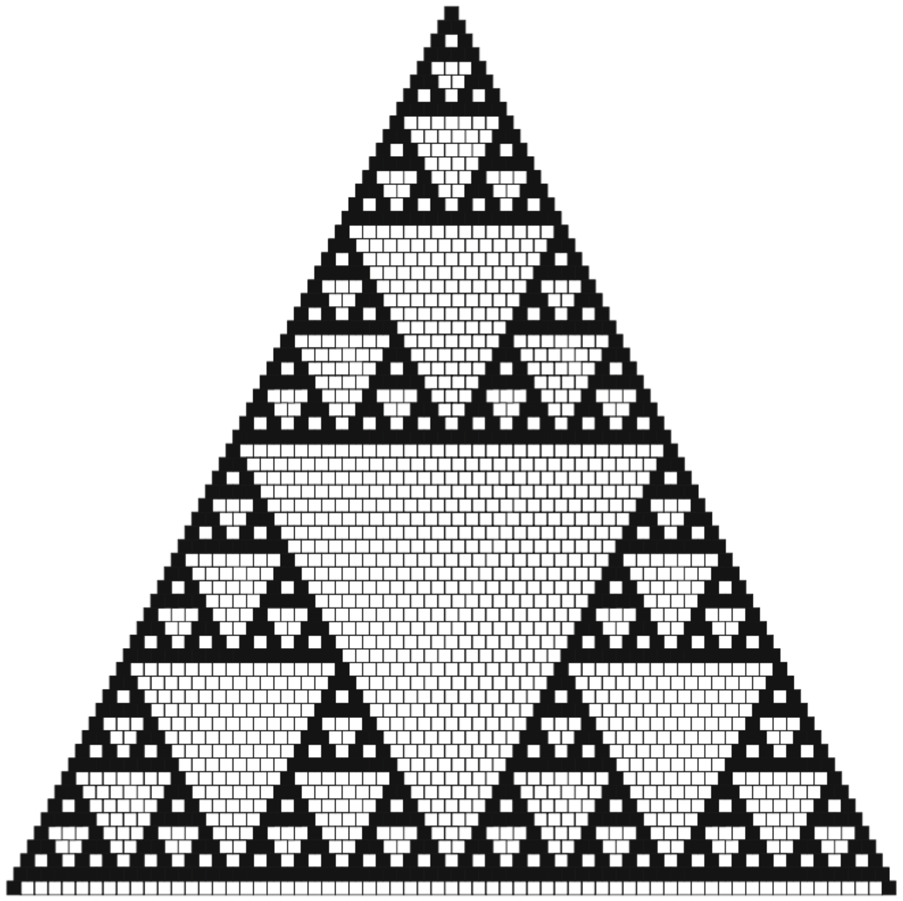
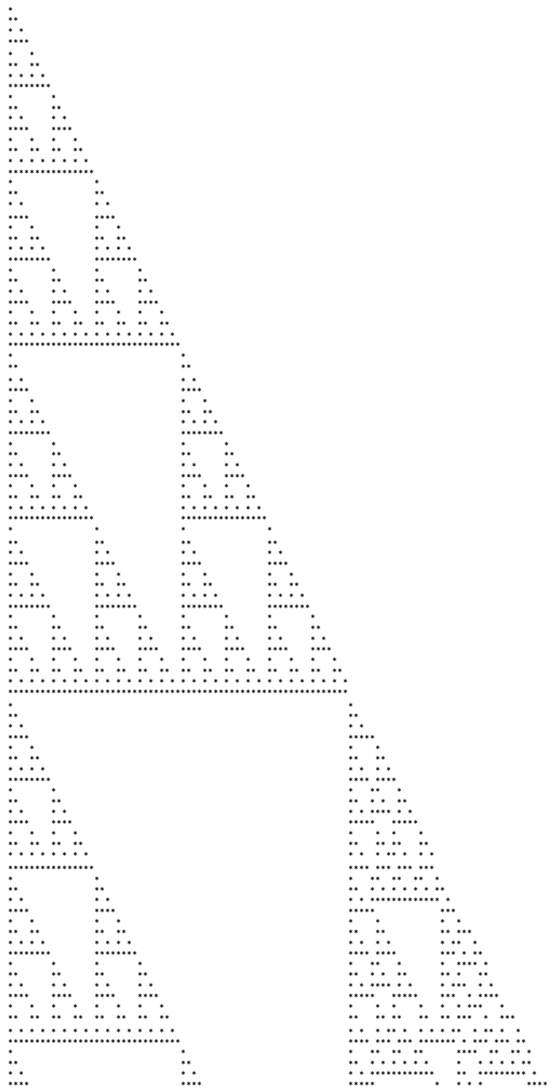


Figure 0.18: Mount Meru mod 2 with 65 lines



Print (meru[i][j]%2?"*":" ”) for 100 rows

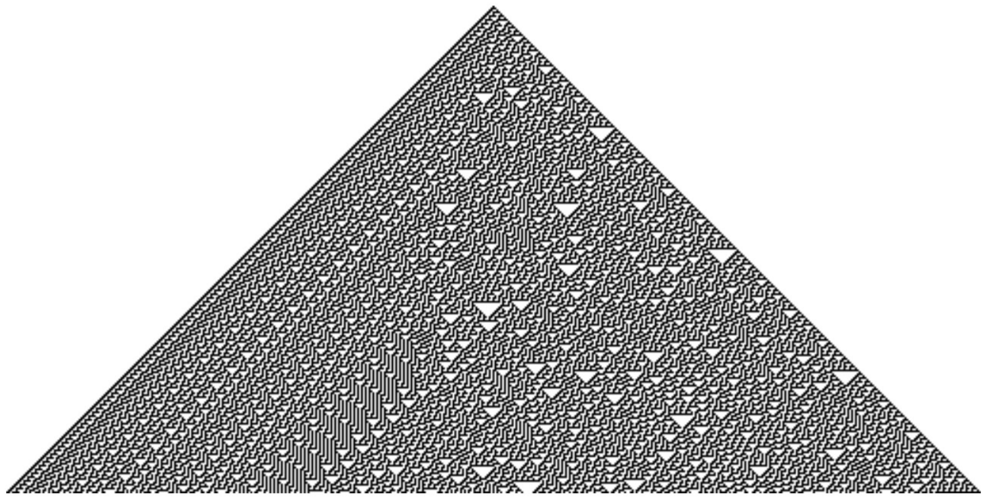
```
#include<stdio.h>
#define ROWS 100

int main() {
  int meru[ROWS][ROWS];

  for (int i=0; i<ROWS; i++) meru[i][0]=1;


  for (int i=0; i<ROWS; i++)
    for (int j=1; j<=i; j++)
      meru[i][j]=meru[i-1][j-1]+meru[i-1][j];


  for (int i=0; i<ROWS; i++) {
    for (int j=0; j<=i; j++)
       printf(meru[i][j]%2? "*":" ");
       printf("\n");
  }
}
```

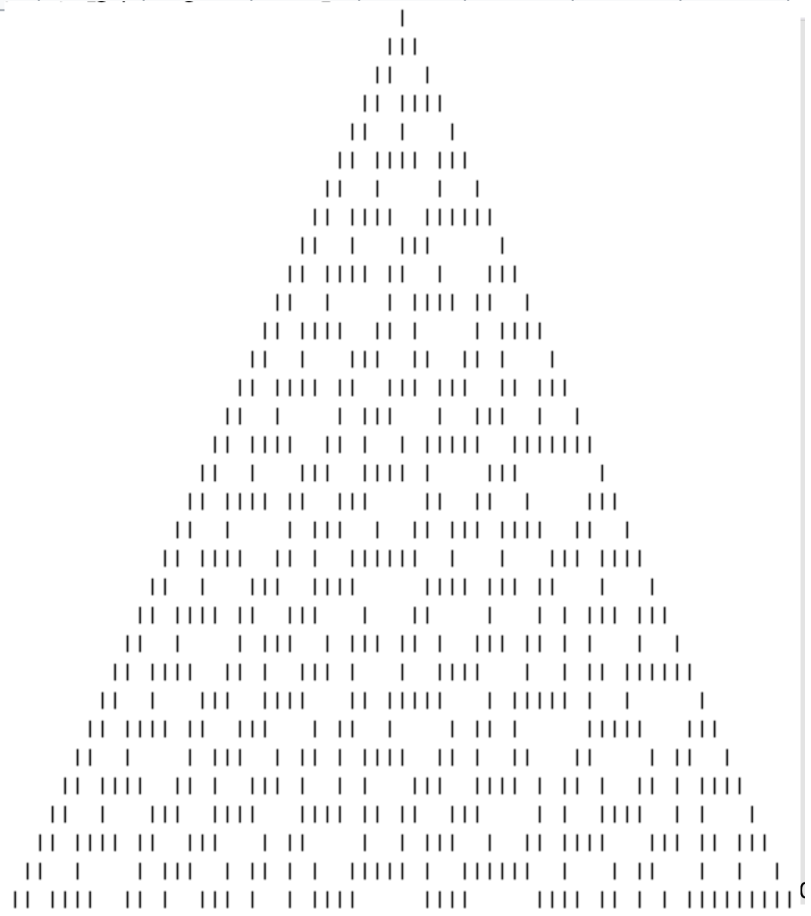
```
#include <stdint.h>
#include <iostream>
int main() {
  uint64_t state = 1u << 31;
  for (int i = 0; i < 32; ++i) {
    for (int j = 64; j--;)
      std::cout << char(state >> j & 1 ? ‘|' : ' ');
    std::cout << '\n';
    state = (state >> 1) ^ (state | state << 1);
  }
}
```

Rule 30: chaotic!

wiki

current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	0	0	1	1	1	1	0


Rule 30 cellular automaton



```
                                |
                               |||
                              ||  |
                             || ||||
                            ||  |   |
                           || |||| |||
                          ||  |    |  |
                         || ||||  ||||||
                        ||  |   |||     |
                       || |||| ||  |   |||
                      ||  |    | |||| ||  |
                     || ||||  || |    | ||||
                    ||  |   |||  ||  || |   |
                   || |||| ||  ||| |||  || |||
                  ||  |    | |||   |  |||  |  |
                 || ||||  || |  | |||||  |||||||
                ||  |   |||  |||| |    |||      |
               || |||| ||  |||    ||  ||  |    |||
              ||  |    | |||  |  || ||| ||||  ||  |
             || ||||  || |  ||||||  |   |   ||| ||||
            ||  |   |||  ||||     |||| ||| ||   |   |
           || |||| ||  |||   |   ||    |   | | ||| |||
          ||  |    | |||  | ||| || |  ||| || | |   |  |
         || ||||  || |  ||| |   |  ||||   |  | || ||||||
        ||  |   |||  ||||   || |||||   | ||||| |  |     |
       || |||| ||  |||   | ||  |    | || |     |||||   |||
      ||  |    | |||  | || | ||||  || |  ||   ||    | ||  |
     || ||||  || |  ||| |  | |   |||  |||| | || |  || | ||||
    ||  |   |||  ||||   |||| || ||  |||    | |  ||||  | |   |
   || |||| ||  |||   | ||    |  | |||  |  || ||||   ||| || |||
  ||  |    | |||  | || | |  ||||| |  ||||||  |   | ||   |  |  |
 || ||||  || |  ||| |  | ||||     ||||     |||| || | | |||||||||
```

Notes

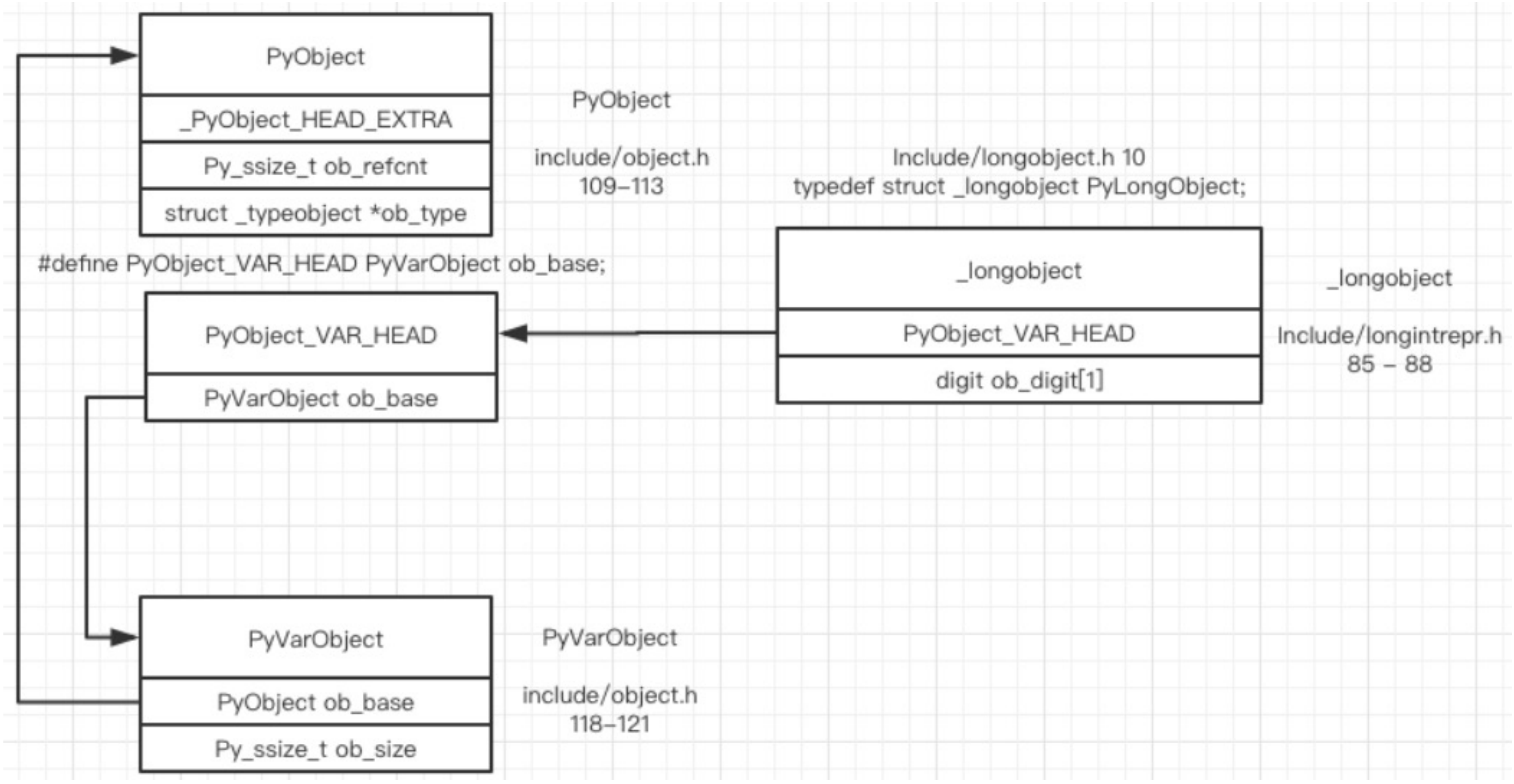
- Python does not include **postfix** operators like the increment (**i++**) or decrement (**i--**) operators available in C.

```
state = 1 << 31
for i in range(32):
  for j in range(64,0,-1):
    if state >> j & 1:
      print ('|',end="")
    else:
      print(' ',end="")
  print("")
  state = (state >> 1) ^ (state | state << 1)
```

- Python does not have the **unsigned right shift operator** denoted by three greater-than signs (>>> in Java).
- This has to do with how Python represents integers internally. Since integers in Python can have an infinite number of bits, the sign bit doesn’t have a fixed position. In fact, there’s no sign bit at all in Python!

Int in CPython

https://github.com/zpoint/CPython-Internals/blob/master/BasicObject/long/long.md



More details:

- 0: **ob_size = 0** indicates that **long object** represents integer 0
  - **ob_digit (16 bits)** field “don’t care”
- 1: **ob_size =** 1 and field in **ob_digit** = 1 with type **unsigned short**
- -1: **ob_size** = -1 and **ob_digit** =1
- 1023: **ob_size** = 1 and **ob_digit** = 1023. Same with 32767 (2^15 -1)
- 32768: **ob_size** = 2 and now **ob_digit** [2] with
  - ob_digit[0]=0
  - ob_digit[1]=1
- 262143 (2^18 -1): **ob_size** = 2 and now **ob_digit** [2] with
  - ob_digit[0]= 7FFF
  - ob_digit[1]= 7

Note: leftmost bit in digit (bit 15)

reserved to handle carry. As integers become large, memory alloc as necessary: eg. adding 1 to 2^30 -1

Rule 110: Turing Complete

wiki

Current pattern	111	110	101	100	011	010	001	000
New state for center cell	0	1	1	0	1	1	1	0

Game of Life also

2D game

1.Any live cell with two or three live neighbours survives.
2.Any dead cell with three live neighbours becomes a live cell.
3.All other live cells die in the next generation.
4.Similarly, all other dead cells stay dead.

The Game of Life is undecidable, which means that given an initial pattern and a later pattern, no algorithm exists that can tell whether the later pattern is ever going to appear.