}

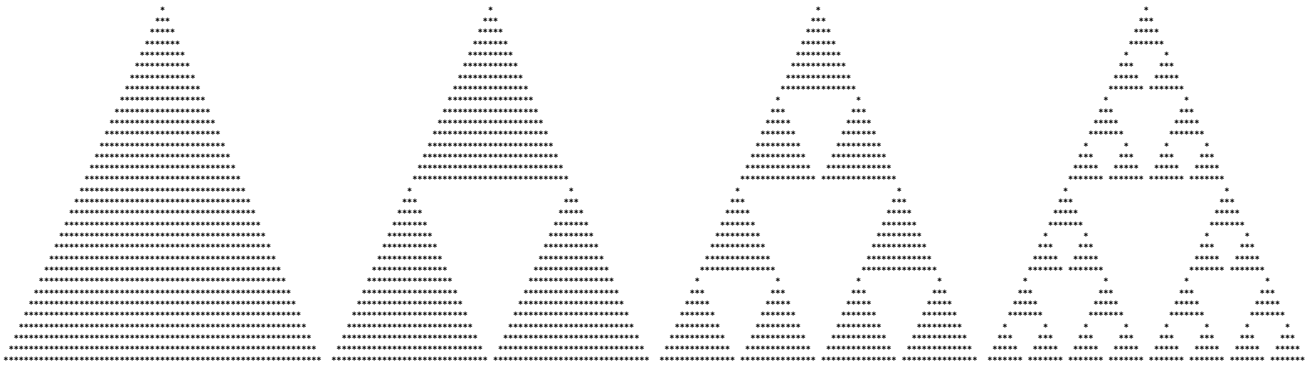
# MC102 - Algoritmos e Programação de Computadores



## ASC ART e Recursão

Nesta tarefa, vamos retornar ao tema ASC ART para exercitarmos recursão. A base para o nosso estudo será o Triângulo de Sierpinski. Ao analisarmos a figura abaixo, fica bem fácil entender o processo de criação deste fractal. A cada passo, um triângulo preto é subdividido em quatro triângulos de tamanho igual, o triângulo central é pintado de branco e os triângulos pretos serão subdivididos no passo seguinte.

Observe agora alguns diagramas em ASC ART. Note que é possível, mesmo com resolução limitada, identificar a estrutura do desenho.



### Elementos do desenho

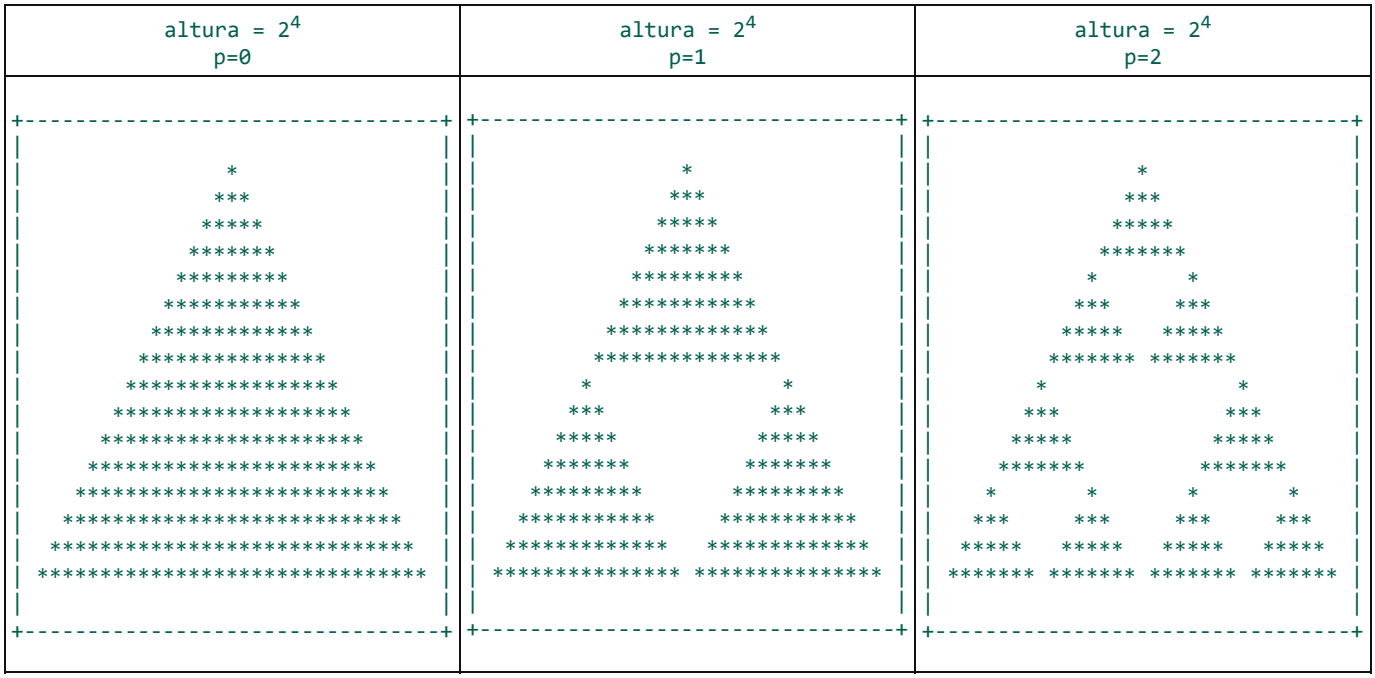
- **Triângulo base:** O desenho é formado a partir de um triângulo base, pintado de preto no desenho original. Em ASC ART, utilizaremos triângulos isósceles e algum caractere para substituir a cor preta. Para facilitar a divisão dos triângulos ao longo dos passos, trabalharemos apenas com triângulos cuja altura seja uma potência de 2 (`altura = 2^N`). A ponta do triângulo terá apenas um caractere, a segunda linha, se houver, terá três caracteres e assim sucessivamente até a base do triângulo, que terá `2 * altura - 1` caracteres. Observe os exemplos abaixo:

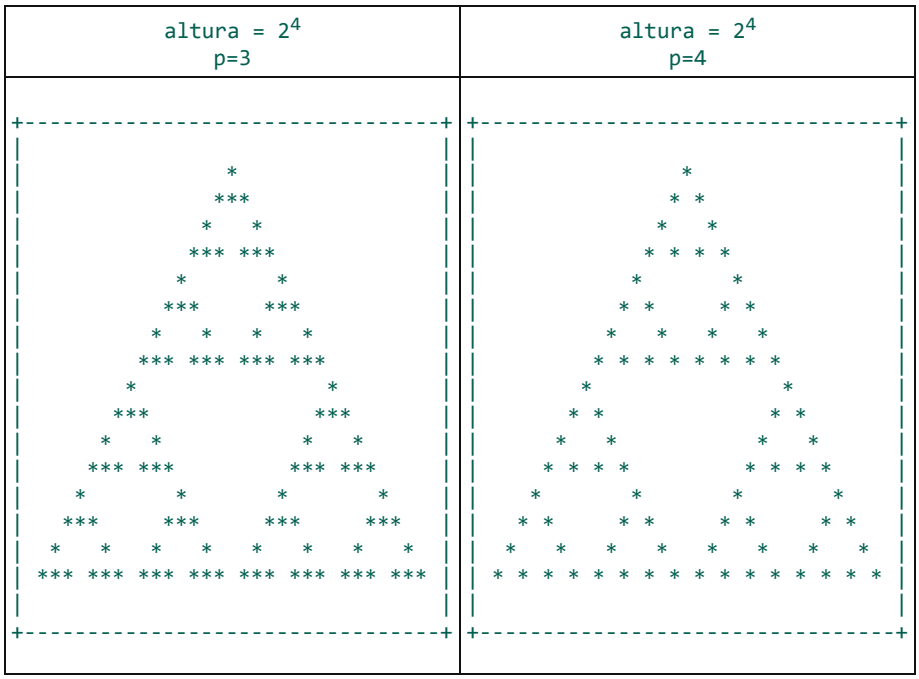
| `altura = 2^0` | `altura = 2^1` | `altura = 2^2` | `altura = 2^3` |
|---|---|---|---|
| `@` | `+`<br>`+++` | `X`<br>`XXX`<br>`XXXXX`<br>`XXXXXXX` | `*`<br>`***`<br>`*****`<br>`*******`<br>`*********`<br>`***********`<br>`*************`<br>`***************` |

- **Triângulos brancos:** Os triângulos brancos terão a mesma forma que o triângulo preto, mas serão desenhados com a ponta para baixo.
- **Tela e moldura:** Os desenhos deverão ser preparados em uma matriz de caracteres, denominada tela, e depois escritos na saída. A tela deverá ser uma matriz retangular com a mesma altura e largura do triângulo base. Ao escrever a matriz, você deverá acrescentar um contorno de caracteres em branco e uma moldura feita com caracteres "-", "|" e "+", como no exemplo abaixo:

```
+-----------------+
|                 |
|        *        |
|       ***       |
|      *****      |
```

- **Profundidade da recursão:** Em um fractal, não há limites para as subdivisões. Nesta tarefa, indicaremos o número de subdivisões, que deve ser sempre menor ou igual ao valor de $\log_2(\texttt{altura})$, sendo `altura` a altura do triângulo base.




## Descrição da entrada

A entrada será formada por três valores:

```
<N>
<p>
<char_preto>
```

A altura do triângulo base e da tela será definida por $2^N$. O inteiro `p` indicará o número de subdivisões e o caractere `<char_preto>` indicará o caractere que preencherá o triângulo base.

## Descrição da saída

A saída será formada por um único desenho de um triângulo isósceles de `altura = 2^N` e `largura = 2 * altura - 1` caracteres do tipo `<char_preto>`. O triângulo deverá apresentar `p` subdivisões do Triângulo de Sierpinski e o desenho deverá estar emoldurado como exemplificado acima.

## Testes com o SuSy

O conjunto de testes será formado por 7 testes abertos e 3 testes fechados. Os testes fechados são variações dos testes abertos em que só foram alterados os caracteres utilizados para desenho. Releia, se necessário, as instruções para fazer os testes em Testes com o SuSy.

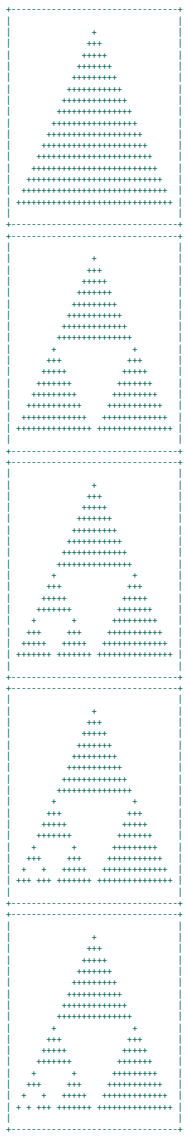
## Dicas para a implementação

Esta tarefa ficará simples de ser implementada se você subdividí-la em problemas menores, que devem ser desenvolvidos e testados separadamente. Veja a sugestão abaixo:

- Desenvolva uma função para criar a `tela` em branco. Você pode usar o comando:

  ```
  tela = [ [" " for j in range(largura)] for i in range(altura)]
  ```

- Desenvolva uma função `print_tela_com_moldura()` para escrever esta `tela` com a moldura solicitada na saída.
- Desenvolva uma função que desenha o triângulo base na `tela`. Teste a saída com a função `print_tela_com_moldura()`.
- Desenvolva uma função que desenha um triângulo branco a partir de coordenadas `x,y` na `tela`. Teste a saída com a função `print_tela_com_moldura()`.
- Organize as chamadas recursivas para o desenho dos triângulos em branco. Durante o desenvolvimento, acrescente chamadas extras à função `print_tela_com_moldura()` de maneira que você possa acompanhar o caminho que está sendo percorrido. Observe os primeiros desenhos para `N = 4, p = 4, char_preto = "+"` em que começamos a subdivisão pelos triângulos inferiores à esquerda:



- Para aumentar sua compreensão sobre as chamadas recursivas, troque a ordem das chamadas e compare os caminhos percorridos.

## Orientações para submissão

Veja aqui a página de submissão da tarefa. O arquivo a ser submetido deve se chamar `lab12.py`. No link Arquivos auxiliares há um arquivo `aux12.zip` que contém todos os arquivos de testes abertos e seus respectivos resultados compactados.

O limite máximo será de 20 submissões. Serão considerados os resultados da última submissão.

O peso desta tarefa é 3.

O prazo final para submissão é 30/11/2019.

A nota desta tarefa é proporcional ao número de testes que executaram corretamente, desde que o código esteja coerente com o enunciado. **A submissão de um código que não implementa o algoritmo requisitado, mas que exibe as saídas esperadas dos testes abertos a partir da comparação de trechos da entrada será considerada fraude e acarretará a atribuição de nota zero à média final da disciplina.**

---

A imagem que ilustra os passos do algoritmo de Sierpinski foi obtida verberte referente ao Triângulo de Sierpinski na Wikipedia. Todos os outros desenhos foram obtidos a partir de programas em Python.