USP - Universidade de São Paulo
Instituto de Ciências Matemáticas e Computação

Introdução à Ciência da Computação I
SCC0221

ASCII Artista

---

1 Introdução

Ada Amaral é uma jovem que tem duas grandes paixões: computação e artes. Ela passava horas em seu celular assistindo a lives de seus artistas preferidos na Twitch e vídeos sobre computação, sempre idealizando o dia que no qual ela teria os próprios equipamentos para fazer as suas artes digitais e programas.

Dados os amores da filha, seus pais resolveram comprar um presente para a menina, porém, por não serem tão abastados, o máximo que conseguiram comprar foi um notebook usado no qual só o teclado funcionava. Ao receber a surpresa, Ada ficou muito empolgada e tomou o problema do presente como algo desafiador e empolgante.

Uma semana de uso e Ada já estava muito a vontade com o uso do computador somente via atalhos no teclado, inclusive fez alguns programinhas introdutórios para seus estudos de computação. Dadas as circunstâncias Ada pesquisou sobre artes digitais feitas apenas com o uso do teclado, e assim a jovem conheceu o mundo da "ASCII Art" — a arte feita com uso apenas de caracteres digitados. Assim, a flexível artista migrou do papel com caneta para o bloco de notas com teclado.

Depois de algum tempo praticando bastante Ada já fazia arte incríveis, especialmente artes com apenas contorno (também conhecidas como "line arts"), e foi nessa época que ela descobriu um concurso que seria a sua oportunidade: "O Concurso de ASCII Artistas" de Capivari (cidade natal da jovem). Concurso o qual tem como premio uma mesa digitalizadora, a qual abriria um mundo gigantesco de possibilidades para Ada.

Entretanto, a jovem artista não poderia submeter suas melhores obras para o concurso, dado que regras do torneio proibiam a submissão de "line arts" para a competição. Mas devido seu outro hobby, Ada teve uma ideia: criar um programa para facilitar a pintura de suas obras de contorno, algo parecido com a ferramenta de baldinho dos softwares para desenho usados pelos idólos de Ada na Twitch.

A computeira artista ainda não é tão boa com códigos e por isso precisa da sua ajuda. Desenvolva um programa em C que, dado o nome do arquivo contendo uma das artes de Ada Amaral, permita que a jovem preencha uma quantidade n de regiões, escolhendo uma cor (um caractere) e um pixel (uma coordenada (x, y) dentro da região) para cada uma dessas regiões, mostrando as etapas, e atualizando o arquivo ao final do código.

2 Entrada

Tem-se como os seguintes dados:

- A primeira linha contém uma string referente ao nome da arte (nome de um arquivo de arte existente) a ser editada;
- A segunda linha contém um inteiro n positivo referente a quantidade de preenchimentos que Ada pretende fazer;
- em seguida têm-se n linhas, cada uma contendo os seguintes dados separados por espaço:
  - Um caractere não branco representante da cor a ser preenchida a região;
  - A coordenada x (que diz com relação às linhas) do pixel inicial de preenchimento;
  - A coordenada y (que diz com relação às colunas) do pixel inicial de preenchimento.

Atenção: a coordenada (0,0) da imagem é referente ao pixel da extrema esquerda superior da imagem.

3 Saída

Dada a entrada, tem-se como saída os seguintes dados:

- "Arte inicial:\n" seguido pela impressão da arte inicial;
- Para cada uma dos n preenchimentos "\nArte após a etapa i:\n" seguido da impressão da arte após cada uma das etapa (i inicia em 0);
- "\nArte enquadrada:\n" seguida da chamada **(!!! após fechar o arquivo !!!)** da função **enquadra_arte(char *nome_do_arquivo_da_arte, int altura_do_quadro, int largura_do_quadro)** — função que não pode ser alterada e que está disponível no runcodes.

4 Requisitos da implementação

Devem ser seguidas as seguintes restrições para a implementação desse trabalho:

- Realizar a **leitura a partir do arquivo** referente a arte a ser editada e carregar essa em uma matriz, fechando o arquivo ao fim da leitura;
- Essa **matriz deve ser alocada dinamicamente (char **)** e **desalocada ao fim de seu uso**, com o uso das funções de alocação e desalocação biblioteca stdlib.h — a função **char *read_line(FILE *stream)** implementada nos exercícios anteriores pode te auxiliar nessa tarefa (lembre-se de liberar a memória ao fim do uso);
- Seu programa deve conter uma **função recursiva a qual realiza o preenchimento na matriz** (essa será melhor explicada na próxima seção);
- Realizar a **escrita no arquivo** (incluindo as quebras de todas as linhas exceto a última — a qual ficará somente com o EOF mesmo) referente a arte editada e carregar essa a partir da matriz, fechando o arquivo ao final do processo;
- Seu programa deve conter uma **boa legibilidade**, isso no geral é contemplado por: **bons nomes de variáveis** (significativos, que dizem o que a variável representa), **indentação correta** (de acordo com escopos criados dadas as necessidades), **boa separação em blocos/modularização do código** e **documentação com comentários**, quando necessários.

5 Implementação da função recursiva para preenchimento

Essa será a função principal do seu programa, e ela segue a seguinte lógica: dada a matriz da arte, o caractere de cor e as coordenadas do pixel inicial de preenchimento (e a cor desse), começando desse pixel escreva o caractere cor em toda a região delimitada pela cor preliminar do pixel inicial. Segue dois esquemas:

```
matriz:
  012345678
 -----------
0| xxxxxxx |
1| x     x |
2| x    x  |
3|  xxxxx  |
 -----------
cor: 'b'
pixel(x, y): (2, 3) -> cor a ser preenchida: ' '

-----------    -----------    -----------    -----------    -----------    -----------
| xxxxxxx |    | xxxxxxx |    | xxxxxxx |    | xxxxxxx |    | xxxxxxx |    | xxxxxxx |
| x     x | -\ | x     x | -\ | x b   x | -\ | xbbb  x | -\ | xbbbb x | -\ | xbbbbbx |
| x    x  | -/ | x b  x  | -/ | xbbb x  | -/ | xbbbbx  | -/ | xbbbbx  | -/ | xbbbbx  |
|  xxxxx  |    |  xxxxx  |    |  xxxxx  |    |  xxxxx  |    |  xxxxx  |    |  xxxxx  |
-----------    -----------    -----------    -----------    -----------    -----------
```

```
matriz:
  012345678
 -----------
0|    .    |
1|    ;    |
2|    x    |
3|     diff|
 -----------
cor: '+'
pixel(x, y): (0, 8) -> cor a ser preenchida: ' '

-----------    -----------    -----------    -----------    -----------    -----------
|    .   +|    |    .  ++|    |    . +++|    |    .++++|    |    .++++|    |    .++++|
|    ;    | -\ |    ;   +| -\ |    ;  ++| -\ |    ; +++| -\ |    ;++++| -\ |    ;++++|
|    x    | -/ |    x    | -/ |    x   +| -/ |    x  ++| -/ |    x +++| -/ |    x++++|
|     diff|    |     diff|    |     diff|    |     diff|    |     diff|    |     diff|
-----------    -----------    -----------    -----------    -----------    -----------
```

Atente-se a alguns detalhes:

- Cuidado para não acessar posições inválidas fora da matriz da arte;
- A função deve preencher apenas pixeis contidos na região de mesma cor da cor preliminar do pixel inicial;
- Após o preenchimento de um pixel sua função deve chamar ela mesma para o preenchimento dos vizinhos desse pixel: os **2 vizinhos horizontais e os 2 vizinhos verticais**;
- Ao final de toda sua recursão sua matriz deve estar alterada como o esperado.

6 Exemplos de entrada e saída

Entrada genérica

```
urso.ascii
2
\ 6 7
/ 6 20
```

urso.ascii inicial

```

  .--.                .--.
 : (\ ". _.......__ ." /) :
  '.     `          `     .'
   /'    _           _    `\
  /      0}         {0      \
 |        /         \        |
 |      /'           `\      |
  \    | .   .==.    . |    /
   '._ \.' \__/  './ _.'
   /   ``'._-''-_.'``   \
```

urso.ascii depois

```

  .--.                .--.
 : (\\". _.......__ ."//) :
  '.\\\\`          `////.'
   /'\\\_          _///`\
  /\\\\\0}        {0/////\
 |\\\\\\\/        \///////|
 |\\\\\/'           `\/////|
  \\\\| .   .==.    . |////
   '._ \.' \__/  './ _.'
   /   ``'._-''-_.'``   \
```

Saída genérica

```
Arte inicial:

  .--.                .--.
 : (\ ". _......_ ." /) :
  '.    `          `    .'
   /'    _          _    `\
  /     0}        {0     \
 |       /        \       |
 |     /'          `\     |
  \    | .  .==.   . |    /
   '._ \.' \__/ './ _.'
   /   ``'._-''-_.'``   \


Arte apos a etapa 0:

  .--.                .--.
 : (\\". _......_ ." /) :
  '.\\\\`          `    .'
   /'\\\_          _    `\
  /\\\\\0}        {0     \
 |\\\\\\\/        \       |
 |\\\\\/'          `\     |
  \\\\| .  .==.   . |    /
   '._ \.' \__/ './ _.'
   /   ``'._-''-_.'``   \


Arte apos a etapa 1:

  .--.                .--.
 : (\\". _......_ ."//) :
  '.\\\\`          `////.'
   /'\\\_          _///`\
  /\\\\\0}        {0/////\
 |\\\\\\\/        \///////|
 |\\\\\/'          `\/////|
  \\\\| .  .==.   . |////
   '._ \.' \__/ './ _.'
   /   ``'._-''-_.'``   \


Arte enquadrada:
              /\

|                              |
|  .--.                .--.    |
| : (\\". _......_ ."//) :    |
|  '.\\\\`          `////.'    |
|   /'\\\_          _///`\     |
|  /\\\\\0}        {0/////\    |
| |\\\\\\\/        \///////|   |
| |\\\\\/'          `\/////|   |
|  \\\\| .  .==.   . |////    |
|   '._ \.' \__/ './ _.'      |
|   /   ``'._-''-_.'``   \    |
|                              |
```

Obs.: saída ficou sem alguns caracteres na parte do enquadramento por falta de suporte da biblioteca feita para gerar a caixa de texto no pdf, mas ela pode ser vista melhor pelas saídas disponíveis no runcodes.

Bom trabalho, obrigado pelo semestre, boas férias :)