INSTITUTO SUPERIOR TÉCNICO

Introdução aos Algoritmos e Estruturas de Dados 2009/2010

Enunciado do 1º Projecto

Data de entrega: 12 de Abril de 2010 às 23h00

1 Introdução

Neste primeiro projecto pretende-se desenvolver um programa cujo objectivo é simular de forma restrita o conhecido jogo `Tetris`. O formato das peças usadas corresponde aos cinco tetraminós na Figura 1. Cada tetraminó tem uma de seis cores: vermelho, amarelo, azul, verde, laranja ou violeta. A Figura 2 mostra os possíveis tetraminós.

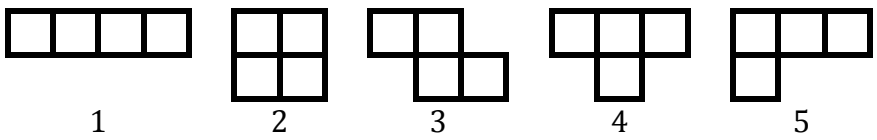

Figura 1: Formato das peças (tetraminós). Os formatos são identificados por inteiros de 1 a 5.

Na versão do jogo correspondente ao 1º projecto assumem-se as seguintes restrições:

- Os tetraminós caem no espaço do jogo sem sofrer deslocações (horizontais para colunas à esquerda ou à direita quando possível) até pararem.
- Apenas podem ser jogados tetraminós com os formatos indicados na Figura 2, isto é, as peças não sofrem rotações ou reflexões.
- Para cada formato de tetraminó na Figura 1 é dada a linha e a coluna onde fica a quadrícula no canto superior esquerdo da peça (ver Figura 3).

- Assume-se que a peça fica sempre dentro do espaço do jogo e sem qualquer sobreposição com outras peças.
- O tamanho do espaço do jogo é fixo: 10 colunas por 20 linhas (ver Figura 4).

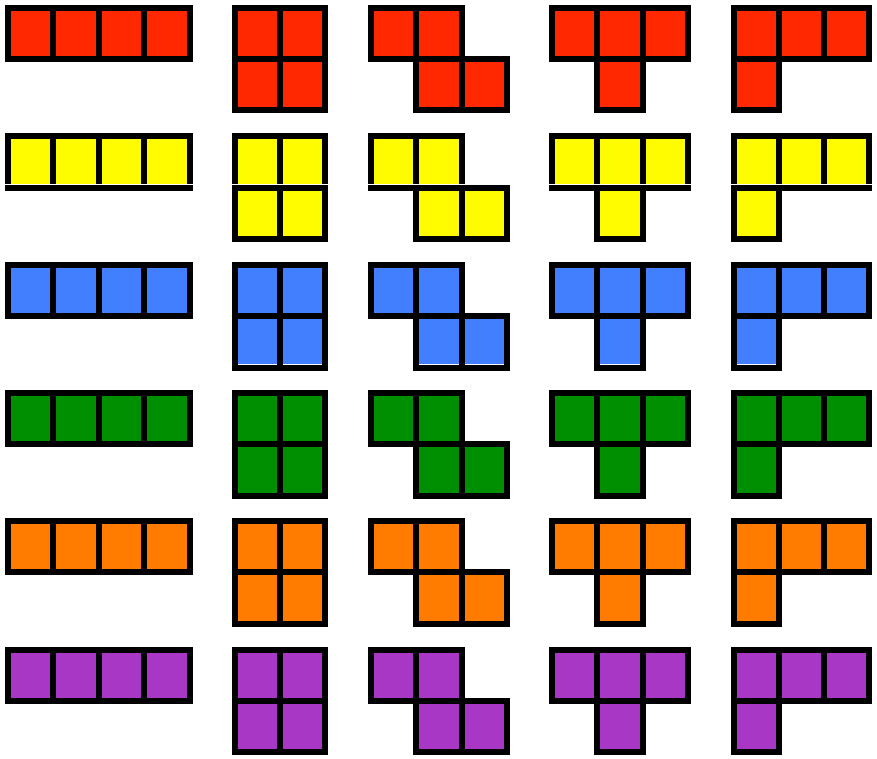

Figura 2: Possíveis tetraminós. As seis cores possíveis são identificadas pelos caracteres R, Y, B, G, O e V.

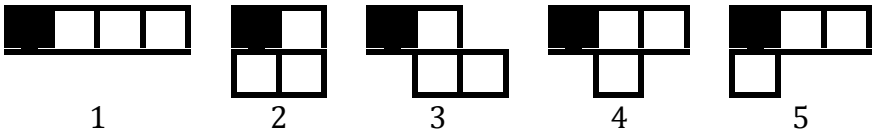

Figura 3: Quadrícula no canto superior esquerdo da peça (a preto) para cada formato de tetraminó na Figura 1.

A Figura 4 mostra um exemplo de final de jogo com vinte tetraminós colocados nas vinte jogadas indicadas na Tabela 1.

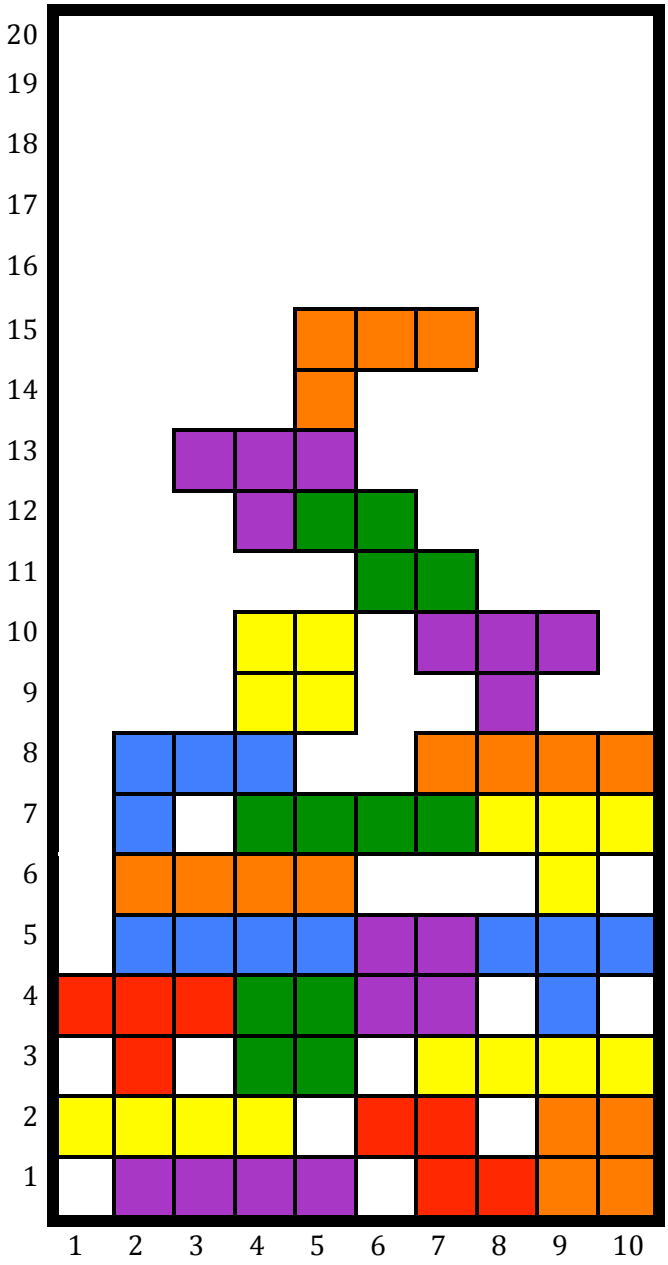

Figura 4: Exemplo de final de jogo.

Tabela 1: Sequência de jogadas efectuadas no jogo da Figura 4.

Jogada	Tetraminó	Linha	Coluna	Cor
1	1	1	2	Violeta (V)
2	3	2	6	Vermelho (R)
3	1	2	1	Amarelo (Y)
4	4	4	1	Vermelho (R)
5	2	2	9	Laranja (O)
6	1	3	7	Amarelo (Y)
7	2	5	6	Violeta (V)
8	2	4	4	Verde (G)
9	1	5	2	Azul (B)
10	1	6	2	Laranja (O)
11	4	5	8	Azul (B)
12	4	7	8	Amarelo (Y)
13	1	7	4	Verde (G)
14	5	8	2	Azul (B)
15	2	10	4	Amarelo (Y)
16	1	8	7	Laranja (O)
17	4	10	7	Violeta (V)
18	3	12	5	Verde (G)
19	4	13	3	Violeta (V)
20	5	15	5	Laranja (O)

2 Especificação do Programa

O programa recebe como *input* uma sequência de tetraminós, aos quais está associada uma linha e uma coluna que permitem determinar a localização final da peça, isto é todas as quadrículas (posições) ocupadas pela peça após a sua queda no espaço do jogo. O valor da coluna indica a coluna onde cai a quadrícula no canto superior esquerdo da peça (ver Figura 3) e não muda durante a queda da peça no espaço do jogo. O valor da linha identifica a linha onde esta quadrícula da peça se localiza quando a peça pára. Por exemplo, se uma peça tem associada a coluna 1 e a linha 2, tal como acontece com o tetraminó amarelo no jogo da Figura 4, então a peça cai encostada ao lado esquerdo do jogo e a sua quadrícula mais à esquerda pára na linha 2.

O valor da coluna e da linha garantem que a peça fica totalmente dentro do espaço de jogo e que as peças não se sobrepõem.

3 Dados de Entrada

O programa deverá ler os dados de entrada a partir do *standard input*. O formato dos dados de entrada será o seguinte:

- uma linha contendo um inteiro N ($N \geq 1$) que denota o número total de peças do jogo;
- uma sequência de N linhas onde cada linha denota uma peça definida da seguinte forma:
  - um inteiro F com valor entre 1 e 5 que define o formato da peça (ver código de cada peça na Figura 1);
  - um espaço em branco;
  - um inteiro L com valor entre 1 e 20 que denota a linha onde a quadrícula no canto superior esquerdo da peça fica no espaço de jogo (ver quadrícula no canto superior esquerdo para cada formato de peça na Figura 3);
  - um espaço em branco;
  - um inteiro C com valor entre 1 e 10 que denota a coluna onde a quadrícula no canto superior esquerdo da peça fica no espaço de jogo;
  - um espaço em branco;
  - uma letra maiúscula M que indica a cor da peça: R, Y, B, G, O ou V.

Assuma que os dados de entrada para o programa não contêm erros sintácticos, isto é, obedecem sempre ao formato descrito nesta secção.

4 Dados de Saída

Após processar as peças como indicado no input, o programa deverá escrever no *standard output* um conjunto de dados sobre a situação final de jogo. A informação a escrever no output será a seguinte e por esta ordem:

- uma linha com um inteiro $LMax$ onde $LMax$ denota a linha de maior índice com peças;
- uma linha com um inteiro $LMin$ onde $LMin$ denota a linha de menor índice do espaço de jogo onde existam peças com o maior número de cores diferentes;
- uma linha em branco;
- uma sequência de 20 linhas que denota a situação final do jogo; Começando na linha de índice 20 e de forma decrescente até à linha de índice 1, temos o seguinte:
  - cada linha é composta pelo número da linha, um espaço em branco, e por 10 caracteres que correspondem ao estado de ocupação de cada coluna dessa linha:

1. quando uma posição (linha,coluna) do jogo estiver ocupada, o caracter a mostrar denota a cor da peça que ocupa esse espaço (R, Y, B, G, O, V);
2. quando uma posição (linha,coluna) do jogo estiver livre, o caracter a mostrar deverá ser o espaço em branco.

5 Exemplo

5.1 Dados de Entrada

Vamos admitir que o ficheiro de entrada, que designaremos por `in.ttr`, contém as jogadas que levam à situação final de jogo ilustrada na Figura 4:

```
20
1  1 2 V
3  2 6 R
1  2 1 Y
4  4 1 R
2  2 9 O
1  3 7 Y
2  5 6 V
2  4 4 G
1  5 2 B
1  6 2 O
4  5 8 B
4  7 8 Y
1  7 4 G
5  8 2 B
2 10 4 Y
1  8 7 O
4 10 7 V
3 12 5 G
4 13 3 V
5 15 5 O
```

5.2 Resultados

Para o ficheiro de entrada `in.ttr` descrito na secção anterior, o programa deverá escrever na saída a seguinte informação:

```
15
4


20
19
18
17
16
15     OOO
14     O
13   VVV
12    VGG
11      GG
10    YY VVV
 9    YY  V
 8  BBB  OOOO
 7  B GGGGYYY
 6  OOOO   Y
 5  BBBBVVBBB
 4 RRRGGVV B
 3  R GG YYYY
 2 YYYY RR OO
 1  VVVV RROO
```

6 Compilação do Programa

Deve concretizar o ficheiro `Makefile` onde deverá definir o processo de geração do executável correspondente ao programa realizado. O compilador a utilizar é o `gcc` com as seguintes opções de compilação: `-ansi -Wall -pedantic`. Este processo deve ser definido na regra `all` do ficheiro `Makefile`. O executável gerado deve ter o nome `proj1`. Para compilar o programa deve executar o seguinte comando:

```
$ make -s all
```

o qual deve ter como resultado a geração do ficheiro executável `proj1`, caso não haja erros de compilação. A execução deste comando não deverá escrever qualquer resultado no terminal (daí a utilização da opção `-s` do comando `make`). Caso a execução deste comando escreva algum resultado no terminal, considera-se que o programa não compilou com sucesso. Por exemplo, durante a compilação do programa, o compilador não deve escrever mensagens de *warning*.

7 Execução do Programa

O programa deve ser executado da forma seguinte:

```
$ ./proj1 < in.ttr > out.txt
```

8 Entrega do Projecto

A entrega do projecto deverá respeitar o procedimento seguinte:

- Na página da disciplina aceda ao sistema para entrega de projectos. O sistema será activado uma semana antes da data limite de entrega. Instruções de como aceder ao sistema serão oportunamente fornecidas.
- Efectue o upload de um ficheiro arquivo com extensão `.tgz` que inclua o seguinte:
  - Ficheiros fonte (`.c`) e cabeçalho (`.h`) que constituem o programa.
  - Ficheiro `Makefile` que permita criar o executável `proj1`.

  Para criar um ficheiro arquivo com a extensão `.tgz` deve executar o seguinte comando na directoria onde se encontram o ficheiro `Makefile` e os ficheiros com extensão `.c` e `.h`, criados durante o desenvolvimento do projecto:

  ```
  $ tar cfz proj1.tgz Makefile *.c *.h
  ```
- Como resultado do processo de upload será informado se a resolução entregue apresenta a resposta esperada num conjunto de casos de teste. Exemplos de casos de teste serão oportunamente fornecidos.
- Data limite de entrega do projecto: **23h00 do dia 12 de Abril de 2010**. Até à data limite poderá efectuar o número de entregas que desejar, sendo utilizada para efeitos de avaliação a **última** entrega efectuada. Deverá portanto verificar cuidadosamente que a última entrega realizada corresponde à versão do projecto que pretende que seja avaliada. Não serão abertas excepções.

9 Avaliação do Projecto

9.1 Componentes da Avaliação

Na avaliação do projecto serão consideradas as seguintes componentes:

1. A primeira componente avalia o desempenho da funcionalidade do programa realizado. Esta componente é avaliada entre 0 e 16 valores.

2. A segunda componente avalia a qualidade do código entregue, nomeadamente os seguintes aspectos: comentários, identação, estruturação, modularidade, abstracção, entre outros. Esta componente poderá variar entre -4 valores e +4 valores relativamente à classificação calculada no item anterior e será atribuída na discussão final do projecto.

Durante a discussão final do projecto será averiguada a participação de cada elemento do grupo na realização do projecto, bem como a sua compreensão do trabalho realizado, sendo a respectiva classificação ponderada em conformidade, isto é, elementos do mesmo grupo podem ter classificações diferentes. Elementos do grupo que se verifique não terem participado na realização do respectivo projecto terão a classificação de 0 (zero) valores.

9.2 Atribuição Automática da Classificação

A classificação da primeira componente da avaliação do projecto é obtida automaticamente através da execução de um conjunto de testes executados num computador com o sistema operativo **GNU/Linux**. Torna-se portanto essencial que o código compile correctamente e que respeite o formato de entrada e saída dos dados descrito anteriormente. Projectos que não obedeçam ao formato indicado no enunciado serão penalizados na avaliação automática, podendo, no limite, ter 0 (zero) valores se falharem todos os testes. Os testes considerados para efeitos de avaliação podem incluir (ou não) os disponibilizados na página da disciplina, além de um conjunto de testes adicionais. A execução de cada programa em cada teste é limitada na quantidade de memória que pode utilizar, até um máximo de 32 MBytes, e no tempo total disponível para execução, sendo o tempo limite distinto para cada teste.

Note-se que o facto de um projecto passar com sucesso o conjunto de testes disponibilizado na página da disciplina não implica que esse projecto esteja totalmente correcto. Apenas indica que passou alguns testes com sucesso, mas este conjunto de testes não é exaustivo. É da responsabilidade dos alunos garantir que o código produzido está correcto.

Em caso algum será disponibilizada qualquer tipo de informação sobre os casos de teste utilizados pelo sistema de avaliação automática. A totalidade de ficheiros de teste usados na avaliação do projecto serão disponibilizados na página da disciplina após a data de entrega.

9.3 Detecção de Cópias

A avaliação dos projectos inclui a utilização de um sistema para detecção de situações de cópia entre projectos. A submissão de um projecto pressupõe o compromisso de honra que o trabalho incluso foi realizado única e exclusivamente pelos alunos referenciados nos ficheiros submetidos para avaliação. A quebra deste compromisso, ou seja a tentativa de um grupo se apropriar de trabalho de outrém (sejam colegas ou outra pessoa), tem como consequência a reprovação de todos os alunos envolvidos (incluindo quem possibilitar a ocorrência de cópias) à disciplina de IAED.

Toda e qualquer situação de fraude ou facilitação de fraude terá então como consequência a reprovação imediata à disciplina de IAED neste semestre, assim como a comunicação da ocorrência ao respectivo Coordenador de curso e ao Conselho Pedagógico do IST para sanções adicionais de acordo com as regras aprovadas pela UTL e publicadas em Diário da República.

9.4 Considerações adicionais

Todos os programas são avaliados do modo seguinte:

```
$ ./proj1 < in.ttr > out.txt; diff out.txt exp.txt
```

em que o ficheiro `exp.txt` representa o resultado esperado da execução do programa para os dados de entrada definidos pelo ficheiro `in.ttr`. A impossibilidade de verificar automaticamente o resultado da execução de um dado programa implica uma penalização de **100%**. Considera-se que um programa passou um teste com sucesso se o resultado produzido por esse programa for exactamente igual ao resultado esperado, o que significa que o comando `diff` não deverá encontrar diferenças entre o resultado produzido pelo programa submetido e o esperado. Para poder ser avaliado, um projecto deverá compilar correctamente num computador com o sistema operativo **GNU/Linux**, sendo o utilizado o compilador **gcc** da **GNU**. A entrega de código não compilável, ou a não inclusão de qualquer dos ficheiros requeridos, ou a utilização de nomes diferentes para o ficheiro executável conduz a uma classificação de 0 (zero) valores. Não serão aceites quaisquer justificações.