

U.C. 21046

Estruturas de Dados e Algoritmos Fundamentais

2025-2026

INSTRUÇÕES

1. Este e-fólio tem uma cotação global de 3 valores.
2. Este e-fólio deve ser submetido em dois locais distintos:
 - A resolução do programa é realizada na plataforma moodle, no recurso VIRTUAL PROGRAMMING LAB e-fólio A, disponibilizado no espaço central da unidade curricular.
 - A resolução deve ainda ser acompanhada por um relatório constituído por um único ficheiro em **formato pdf**, entregue única e exclusivamente através do recurso “E-Fólio A”, da turma respetiva.
3. O nome do ficheiro PDF deve seguir a norma “eFolioA” + <nº estudante> + <primeiro nome> + <último nome>.
4. Não são aceites e-fólios manuscritos, i.e. tem penalização de 100%.
5. Na primeira página do e-fólio deve constar o nome completo do estudante bem como o seu número. Penalização de 10% a 100%.
6. Durante a realização do e-fólio, os estudantes devem concentrar-se na resolução do seu trabalho individual, não sendo permitida a colocação de perguntas ao professor ou entre colegas.
7. A interpretação das perguntas também faz parte da sua resolução, se encontrar alguma ambiguidade deve indicar claramente como foi resolvida.

ENUNCIADO

Pretende-se desenvolver um programa em linguagem C++11 padrão que aceite comandos para a gestão de duas estruturas de dados sequenciais de inteiros: uma pilha (LIFO) e uma fila (FIFO), ambas implementadas manualmente com recurso a listas simplesmente ligadas.

Os itens armazenados são números inteiros (positivos ou negativos), representando simultaneamente o papel de chave e de informação.

A descrição das especificações, desenvolvimento e teste do programa são realizados na plataforma moodle, no recurso VIRTUAL PROGRAMMING LAB e-fólio A, disponibilizado no espaço central da unidade curricular.

As estruturas são inicializadas por defeito como vazias e são alteradas com comandos especificados num ficheiro de entrada fornecido ao programa pela entrada padrão (stdin em C e cin em C++), um comando por linha, podendo um comando ter vários argumentos, com o seguinte formato,

`cmd arg0 arg1 ...`

onde “cmd” indica o nome do comando a executar, “arg” é um tipo de argumento do comando e “...” a seguir a um argumento indica que este pode ser repetido várias vezes indefinidamente.

Na descrição dos comandos, um argumento pode ser representado por: “item” indicando um inteiro; “qtd” indicando uma quantidade inteira maior ou igual a zero.

A lista dos comandos a implementar é a indicada a seguir. No caso de mensagens de saída de dados, o seu formato é indicado em estilo da linguagem C.

Operações sobre a pilha

Comando	Descrição
<code>stack_push item...</code>	Insere itens na pilha pela ordem apresentada. O último item indicado deve ficar no topo da pilha.
<code>stack_pop</code>	Remove o item do topo da pilha.
<code>stack_top</code>	Imprime o item do topo da pilha. Formato “Pilha(top)= %d\n”.
<code>stack_print</code>	Imprime toda a pilha, do topo para a base. Formato “Pilha= %d %d ... \n”.
<code>stack_dim</code>	Imprime o número total de itens na pilha. Formato “Pilha tem %d itens\n”.
<code>stack_clear</code>	Remove todos os nós da pilha.

Operações sobre a fila

Comando	Descrição
<code>queue_enqueue item...</code>	Insere itens no fim da fila pela ordem apresentada.
<code>queue_dequeue</code>	Remove o item da frente da fila.
<code>queue_front</code>	Imprime o item da frente da fila. Formato “Fila(front)= %d\n”.
<code>queue_back</code>	Imprime o item do fim da fila. Formato “Fila(back)= %d\n”.
<code>queue_print</code>	Imprime toda a fila, da frente para o fim. Formato “Fila= %d %d ... \n”.
<code>queue_dim</code>	Imprime o número total de itens na fila. Formato “Fila tem %d itens\n”.
<code>queue_clear</code>	Remove todos os nós da fila.

Comandos específicos

Comando	Descrição
<code>transfer_stack_queue qtd</code>	Move qtd itens da pilha para a fila, retirando sempre do topo da pilha e inserindo no fim da fila, um a um.
<code>reverse_queue</code>	Inverte a ordem dos itens atualmente existentes na fila, recorrendo obrigatoriamente a uma pilha auxiliar implementada pelo próprio estudante.

Exemplo: se a pilha contiver, do topo para a base, “7 5 2” e a fila contiver, da frente para o fim, “10 11”, então após o comando:

`transfer_stack_queue 2`

a pilha passa a conter “2” e a fila passa a conter “10 11 7 5”.

Todos os comandos da pilha que não possam ser executados devido à pilha estar vazia devem imprimir uma mensagem com o formato “Comando %s: Pilha vazia!\n”.

Todos os comandos da fila que não possam ser executados devido à fila estar vazia devem imprimir uma mensagem com o formato “Comando %s: Fila vazia!\n”.

Isto inclui os comandos `stack_clear` e `queue_clear` quando a respetiva estrutura se encontrar vazia.

O comando `transfer_stack_queue` deve imprimir uma mensagem com o formato “Comando %s: Quantidade invalida!\n” sempre que qtd seja inválida ou superior ao número de itens atualmente existentes na pilha.

Projete as estruturas de dados (classes) adequadas ao programa que se pretende desenvolver. No que respeita aos atributos, o nó deve possuir o item armazenado e o apontador para o nó seguinte. A pilha deve possuir um apontador para o topo e um contador do número de nós. A fila deve possuir apontadores para o primeiro e último nós e um contador do número de nós. Todos os nós e estruturas utilizadas no programa devem ser instâncias de classes (objetos) e não um conjunto de variáveis soltas ou isoladas.

No desenvolvimento do programa não é permitido utilizar algoritmos com nós falsos (dummy nodes). Os métodos (funções membro) são livres, mas o código dos métodos deve ser definido fora das classes, que apenas devem ter os respetivos protótipos. Os métodos e os comandos devem ser implementados tendo em conta a sua eficiência.

O ficheiro de entrada pode ter linhas em branco e o número de espaços que separa o comando e os argumentos pode ser qualquer. Também podem existir linhas de comentário, caso em que começam obrigatoriamente pelo carácter ‘#’ na 1ª posição. O programa deve ler e processar o ficheiro de entrada uma linha inteira de cada vez, num ciclo ler-linha processar-linha até chegar ao fim do ficheiro. É estritamente proibido ler o ficheiro de entrada todo de uma vez para memória.

No desenvolvimento do programa em C++11 não devem ser utilizadas variáveis globais nem a STL para as estruturas de dados e algoritmos estudados. Restrições aplicam-se, nomeadamente, aos includes `<vector>`, `<array>`, `<deque>`, `<forward_list>`, `<list>`, `<map>`, `<queue>`, `<set>`, `<stack>`, `<unordered_map>`, `<unordered_set>` e em parte de `<algorithm>`. Não devem ser usados smart pointers.

Exemplo de dados entrada / saída padrão

Input:

```
# Esta linha é um exemplo de comentário
stack_push 4 7 -2
stack_print
queue_enqueue 10 20
transfer_stack_queue 2
queue_print
stack_print
```

Output:

```
Pilha= -2 7 4
Fila= 10 20 -2 7
Pilha= 4
```

Atenção: todas as linhas de saída são terminadas por \n sem espaços adicionais. No painel “Comments” podem não ser visíveis espaços adicionais da saída do programa, o que pode levar a pensar que o output está correto. No caso de desenvolver código em ambiente externo ao VPL (por exemplo Windows), para compatibilidade utilize codificação UTF-8 no editor de texto.

Critérios de correção

- O programa desenvolvido difere significativamente das especificações e instruções do enunciado => 0 valores.
- O programa utiliza variáveis globais ou componentes da STL não permitidas no enunciado do VPL => 0 valores.
- O programa altera o nome das variáveis e classes definidas nas templates de código do VPL => 0 valores.
- O programa não compila ou produz avisos com g++ -Wall -std=c++11 => 0 valores.
- O código do programa não está correta e uniformemente indentado de modo a permitir a sua leitura fácil => 0 valores.
- O programa não está comentado => 0 valores. Os comentários no programa devem elucidar questões relevantes do código locais ao comentário e não o funcionamento geral do programa, que deve ser explicado no relatório.
- O programa é avaliado tendo como ponto de partida a percentagem de testes com resultado positivo. O nível de simplicidade e qualidade do código também é avaliado. Programas considerados mal estruturados, demasiado complexos, confusos ou ineficientes podem ser penalizados até 50%.
- Apenas são considerados os resultados e o código dos programas constantes no recurso VPL.
- **O e-fólio só é considerado entregue com a submissão do relatório em formato pdf no respetivo recurso no espaço turma.** O relatório tem formato livre, com um máximo de 2 páginas úteis além da capa/cabeçalho, com letra mínima 11. **Deve explicar sucintamente o funcionamento do programa, os pontos principais e opções tomadas, e indicar a complexidade assintótica das operações** `stack_push`, `stack_pop`, `queue_enqueue`, `queue_dequeue`, `transfer_stack_queue` e `reverse_queue`.
- Trabalhos com plágio comprovado ou muito similares entre si, independentemente da origem, serão anulados.

Nota ética: Nunca é de mais referir que o código a apresentar como solução para este e-fólio deve ser 100% original do aluno. A probabilidade de duas pessoas que efetivamente não comunicaram entre si apresentarem programas “quase iguais” é considerada nula. Isto é válido para qualquer par de alunos, assim como entre um aluno e qualquer outra pessoa, em particular através da Internet, onde existem inúmeras soluções e código para os mais variados problemas, em sites, fóruns, blogs, IA, etc.

Bom trabalho!

FIM