UNIVERSIDADE DO ESTADO DO AMAZONAS
ESCOLA SUPERIOR DE TECNOLOGIA
NÚCLEO DE COMPUTAÇÃO - NUCOMP

ALGORITMOS E ESTRUTURAS DE DADOS I 2017/1
PROF. FLÁVIO JOSÉ MENDES COELHO
15/05/2017

PROJETO PRÁTICO 1

1 Objetivos

Este **Projeto Prático 1 – PP1**, tem o objetivo de exercitar e avaliar suas habilidades em:

- Codificar os tipos abstratos de dados LISTA, PILHA E FILA na linguagem de programação exigida neste enunciado, e solucionar problemas relacionados a estas estruturas de dados;
- Demonstrar seu domínio sobre o código que você desenvolveu no **PP1** e mostrar que sabe realizar modificações locais neste código à pedido de um avaliador;
- Apresentar argumentos lógicos, razoáveis, para questões práticas ou teóricas levantadas por um avaliador sobre o seu **PP1**.

2 Descrição do problema

Sua equipe deve escrever um interpretador para a linguagem de programação Ni[1]. A linguagem é imperativa e é descrita como segue:

PROGRAMA-NI: é uma sequência de definições de funções.

FUNÇÃO-NI: é iniciada com um nome formado apenas por uma única letra maiúscula (sem parâmetros), seguida espaço, seguida de ":" e quebra de linha, seguida de uma sequência de COMANDOS-NI (indentados como em Python, e um comando por linha), terminada com uma linha em branco. Forma geral:

```
NOME-DA-FUNÇÃO-NI :
    COMANDO-NI_1
    COMANDO-NI_2
    ...
    COMANDO-NI_n
    LINHA-EM-BRANCO
```

[1]Fruto do esforço inenarrável da mente iluminável do horronorável prof. Kaninchen.

COMANDO-NI: é um dos dois seguintes comandos: (1) CHAMADA-DE-FUNÇÃO-NI; (2) IMPRESSÃO-DE-INTEIROS.

CHAMADA-DE-FUNÇÃO-NI: escreve-se apenas o nome da função chamada. O fluxo de execução salta para a função chamada e continua a partir do primeiro comando desta função até alcançar seu final (linha em branco). Ao término da execução da função chamada, o fluxo de execução retorna para a função chamadora, e continua o processamento a partir da linha seguinte à linha da chamada da função.

IMPRESSÃO-DE-INTEIROS: inicia com a palavra reservada `PRINT`, seguido de espaço, seguido de um argumento que deve ser um número inteiro não-negativo. O comando imprime o valor do argumento na tela seguido de um espaço em branco.

FUNÇÃO-NI-INICIAL: é uma função especial que inicia o processamento de um PROGRAMA-NI. Esta função chama-se `Z` (semelhante à função `main` em C/C++).

Abaixo, segue um exemplo de um código-fonte **Ni**.

```
B :
  PRINT 5
  A
  PRINT 6

A :
  PRINT 3
  PRINT 4

Z :
  PRINT 1
  A
  B
  PRINT 2
```

A execução deste código-fonte imprimirá a sequência de números `1 3 4 5 3 4 6 2`. O interpetador **Ni** deverá tomar um programa **Ni** como entrada e executar seus comandos um a um, em sequência, iniciando o processamento pela função `Z`. Considere associar cada linha de código a um pseudo-endereço de memória. Isto corresponde a armazenar cada linha de código em uma lista com implementação estática. Quando uma função-**Ni** `A` chamar outra função `B`, o pseudo-endereço da linha seguinte à linha da chamada de `B` será guardado. Quando o processamento em `B` terminar, o interpretador fará o processamento retornar para `A` recuperando o pseudo-endereço guardado anteriormente. Uma pilha com implementação dinâmica é uma estrutura de dados eficiente para tratar este processo de chamada à funções e seu retorno às funções chamadoras.

3 Entradas e saídas do problema

Entradas. O **run.codes** é um *juiz online*, isto é, um site que realiza uma "correção" automática do seu programa. O processo é o seguinte: (1) você submete seu programa ao run.codes;

(2) o run.codes compila e testa-o para um conjunto de pares de entrada/saída. Cada par destes se chama um **caso de teste**. Ao ser executado para um dado caso de teste, se seu programa ler corretamente a entrada do caso de teste e produzir como saída a mesma saída do caso de teste, então o run.codes atribuirá um ponto (1,0) para sua equipe. Caso contrário, o run.codes atribuirá zero pontos (0,0), e indicará se a saída do seu programa não casa com a saída esperada do caso de teste, ou se houve algum erro de compilação. Para este projeto haverá 10 casos de testes.
Uma entrada de um caso de teste do run.codes será uma string grande contendo todo o código-fonte de um programam **Ni**. Seu programa deverá ler cada linha desta entrada copiando cada uma para uma string interna do seu programa. O interpretador Ni iniciará o processamento a partir desta string interna.

Saída. A saída é a sequência correta de impressão dos números inteiros (separados por espaço em branco, e terminada com espaço em branco) impressos pelo programa **Ni**. Por exemplo, o programa **Ni** acima imprimirá a seguinte saída:

```
0 2 1 3 4 2 1 3 5 6
```

Por exemplo, se a entrada de um caso de teste fosse o programa **Ni** acima, então a saída do caso de teste seria `0 2 1 3 4 2 1 3 5 6 ` , e seu interpretador deveria gerar exatamente esta mesma saída.

4 Requisitos do projeto

1. **Equipes**. Este projeto deve ser desenvolvido por uma equipe de dois ou três estudantes. Não serão aceitas equipes com número menor ou maior de participantes.

2. **Ferramentas e técnicas**. O projeto deve ser codificado em C++. Será permitido programação procedural, mas todos as estruturas de dados (TADS) deverão ser programadas orientadas a objeto, generalizadas (templates) e encapsuladas.

3. **Padrões de codificação**. Siga o *CamelCase* do Java como padrão de nomeação de identificadores. A indentação e posisionamento de chaves deve seguir o padrão K&R. variante de Stroustrup (`en.wikipedia.org/wiki/Indent_style#K.26R_style`).

4. **Compilador**. Indica-se o uso do compilador *GCC - the GNU Compiler Collection* (http://gcc.gnu.org), ou sua variante *Mingw* para para Microsoft Windows.

5. **Submissão**. Todo o projeto deverá ser submetido ao juiz online em um único arquivo fonte com extensão `cpp`.

6. **Bibliotecas e funções**. Sua equipe não deve utilizar nenhuma estrutura de dados pronta (listas, pilhas, filas, vetores, grafos, etc) da *Standard Template Library* - STL, ou de qualquer outra biblioteca C++. É suficiente o uso de `iostream` e `cstdlib` (para uso de qualquer outro arquivo de cabeçalho, fale com o professor). O uso de estruturas de dados prontas ou funções de bibliotecas conforme explicado acima, implicará na atribuição da nota mínima ao projeto.

5 Pontuação

Os pontos do projeto correspondem aos pontos obtidos no run.codes. Porém, será verificado se o projeto não constitui plágio. A constatação de plágio implica na atribuição automática da nota mínima para o projeto, e portanto para cada membro da equipe.

6 Datas

- Emissão deste enunciado: 20/05/2017 às 12h30min (hora local).
- Abertura do juiz online: 22/05/2017 às 6h (hora local).
- Fechamento do juiz online: 02/05/2017 às 23h59min (hora local).

As datas podem ser modificadas para atender demandas da turma ou do professor. Neste caso, as mudanças serão comunicadas ao inscritos no grupo de AED1 por e-mail.

CÓDIGO DE ÉTICA

Este projeto é uma avaliação acadêmica e deve ser concebido, projetado, codificado e testado pela equipe, com base nas referências fornecidas neste enunciado ou nas aulas de Algoritmos e Estruturas de Dados, ou por referências indicadas pelo professor, ou com base em orientações do professor para com a equipe por solicitação desta. Portanto, não copie código pronto da Internet para aplicá-lo diretamente a este projeto, não copie código de outras equipes, não forneça seu código para outras equipes, nem permita que terceiros produzam este projeto em seu lugar. Isto fere o código de ética desta disciplina e implica na atribuição da nota mínima ao trabalho.

Referências

[1] COELHO, Flávio. Slides das aulas de *Algoritmos e Esruturas de Dados I*. Disponível em https://est.uea.edu.br/fcoelho. Universidade do Estado do Amazonas, Escola Superior de Tecnologia, Núcleo de Computação - NUCOMP. Semestre letivo 2016/1.

[2] C++. In: *WIKIPÉDIA, a enciclopédia livre*. Flórida: Wikimedia Foundation, 2016. Disponível em: <https://pt.wikipedia.org/w/index.php?title=C%2B%2B&oldid=45048480>. Acesso em: 17 abr. 2016.

[3] C++. In: *cppreference.com*, 2016. Disponível em <http://en.cppreference.com/w/>. Acesso em: 17 abr. 2016.

[4] CORMEN, T. H., Leiserson, C. E., Rivest, R. L., Stein C. *Introduction to Algorithms*, 3rd edition, MIT Press, 2010

[5] KNUTH, Donal E. *Fundamental Algorithms*, 3rd.ed., (vol. 1 de The Art of Computer Programming), Addison-Wesley, 1997.

[6] KNUTH, Donal E. *Seminumerical Algorithms*, 3rd.ed., (vol. 2 de The Art of Computer Programming), Addison-Wesley, 1997.

[7] KNUTH, Donal E. *Sorting and Searching*, 2nd.ed., (vol. 3 de The Art of Computer Programming), Addison-Wesley, 1998.

[8] STROUSTRUP, Bjarne. *The C++ Programming Language*. 4th. Edition, Addison-Wesley, 2013.

[9] STROUSTRUP, Bjarne. *A Tour of C++*. Addison-Wesley, 2014.

[10] SZWARCFITER, Jayme Luiz et. alii. *Estruturas de Dados e seus Algoritmos*. Rio de Janeiro. 2a. Ed. LTC, 1994.

[11] WIRTH, Niklaus. *Algoritmos e Estruturas de Dados*. Rio de Janeiro. 1a. Ed. Prentice - Hall do Brasil Ltda., 1989.

[12] ZIVIANI, Nívio. *Projeto de Algoritmos com Implementação em Java e C++*. 2a. Edição. Cengage Learning, 2010.

[13] ZIVIANI, Nívio. *Projeto de Algoritmos com Implementação em Pascal e C*. 3a. Ed. São Paulo: Cengage Learning, 2012.