Universidade Federal de Ouro Preto
ICEA/DECSI

Disciplina: AEDS I

Trabalho Prático 3 – TAD

Duração: até 2 semanas – 10 Pontos

1 Objetivos

O objetivo desse trabalho é rever conceitos básicos de programação bem como explorar os conceitos de Tipos Abstratos de Dados (TADs) e análise de complexidade.

2 Descrição

Você deverá implementar um TAD chamado **TConj** para representar conjuntos de elementos inteiros. Seu tipo abstrato deverá armazenar os dígitos do conjunto e o número de elementos n. Considere que o tamanho máximo de um conjunto é limitado pela memória principal (você deverá usar *alocação dinâmica*) e deve crescer de 10 em 10 elementos (consulte a função em C `realloc` não será permitido o uso de encadeamento). Use arranjos unidimensionais (vetores) para implementar o TAD. Por exemplo, considerando que o seu conjunto armazena os elementos 5, 2, 3, 9, e 1, seu TAD deverá ter a seguinte estrutura:

TConj

Elementos	5	2	3	9	1	...
Índices	0	1	2	3	4	...

N	5

Lembre-se ainda que segundo a definição de conjuntos os elementos são únicos, ou seja, seu TAD deve impedir a existência de elementos repetidos.

As operações (interfaces) que devem ser realizadas em seu TAD são:

1.) Inicializar o conjunto: cria um conjunto vazio (já com 10 elementos físicos).

```
void inicializar(TConj* pA);
```

2.) Inserir um elemento no conjunto, retornando 1 se possível e 0 caso contrário.

```
int inserir_elemento(TConj* pA, int elem);
```

3.) Excluir um elemento do conjunto, retornando 1 se realizado e 0 caso contrário.

```
int excluir_elemento(TConj* pA, int elem);
```

4.) Atribuir ("Setar") um elemento em uma posição do conjunto, retornando em 1 caso de sucesso e 0 em caso contrário.

```
int set_elemento(TConj* pA, int elem, int pos);
```

5.) Recuperar elemento de uma posição do conjunto, retornando em 1 caso de sucesso e 0 em caso contrário. Obs: note que a passagem `pelem` é por referência.

```
int get_elemento(TConj A, int pos, int* pelem);
```

6.) Testar se um elemento está presente no conjunto, retornando sua posição ou -1 caso o não esteja presente.

```
int testar_elemento(TConj A, int elem);
```

7.) Gerar um conjunto aleatoriamente com `n` elementos.

```
TConj gerar_conjunto(int n);
```

8.) Criar um conjunto a partir de um número. Ex: o número 2537 gera o conjunto {2,5,3,7} com `n = 4`.

```
TConj num2conj(int num);
```

9.) Gerar um número a partir do conjunto. (Operação inversa da descrita acima)

```
int conj2num(TConj A);
```

10.) Comparar 2 conjuntos, retornando 1 se são iguais e 0 se são diferentes.

```
int comparar(TConj A, TConj B);
```

11.) Imprimir os elementos do conjunto.

```
void imprimir(TConj A);
```

12.) Unir os conjuntos A e B, gerando um terceiro conjunto C.

```
TConj uniao(TConj A, TConj B);
```

13.) Fusionar (intersecção) os elementos do conjunto A e B, gerando um terceiro conjunto C.

```
TConj inter(TConj A, TConj B);
```

14.) Subtrair o conjunto A do conjunto B, gerando um terceiro conjunto C.

```
TConj subtrair(TConj A, TConj B);
```

Implemente o seu TAD em arquivos separados do programa principal (TConj.c e TConj.h). Se necessário, você pode criar outras funções auxiliares em seu TAD. Suas funções devem executar testes de consistência (por exemplo, não se pode setar um elemento na posição 8 se o conjunto foi configurado para 4 elementos).

Uma vez criado o seu TAD, você deverá testado no programa de testes mostrado a seguir:

```
1  main() {
2    TConj A, B, C, D;
3    int elemento, num, i;
4    inicializar(&A);
5
6    srand(171);
7    for (i = 0; i < 20; i++) inserir_elemento(A, rand() % 40);
8    imprimir(A);
```

```
9
10    srand(171);
11    for (i = 0; i < 10; i++) excluir_elemento(&A, rand() % 20 - 1);
12    imprimir(A);
13
14    if (testar_elemento(A, 9) == -1)
15      printf("o elemento 9 nao esta presente em A");
16    B = gerar_conjunto(10);
17    num = conj2num(B);
18    C = num2conj(num);
19    if (comparar(B, C)) printf("os conjuntos sao iguais\ n");
20    else printf("os conjuntos sao diferentes\ n");
21
22    D = uniao(A, B);
23    imprimir(D);
24    D = inter(A, B);
25    imprimir(D);
26    D = subtrair(A, B);
27    imprimir(D);
28  }
```

3 O que deve ser entregue?

- Código fonte do programa em C (bem identada e comentada).
- Documentação do trabalho seguindo o padrão da SBC (Template SBC.). **LIMITE SUA DOCUMENTAÇÃO A NO MÁXIMO 7 PÁGINAS.** Entre outras coisas, a documentação deve conter:

  1.) Introdução: descrição do problema a ser resolvido e visão geral sobre o funcionamento do programa.

  2.) Implementação: descrição sobre a implementação do programa. Deve ser detalhada a estrutura de dados utilizada (de preferência com diagramas ilustrativos), o funcionamento das principais funções e procedimentos utilizados, o formato de entrada e saída de dados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado.

  3.) Estudo de complexidade: estudo da complexidade do tempo de execução dos procedimentos implementados e do programa como um todo (notação O).

  4.) Listagem de testes executados: os testes executados devem ser simplesmente apresentados.

  5.) Conclusão: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.

  6.) Bibliografia: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da Internet se for o caso.

  7.) Formato: obrigatoriamente em **PDF**

4 Dicas

- Consulte as dicas do Prof. Nívio Ziviani de como deve ser feita uma boa implementação e documentação de um trabalho prático (LINK).
- Clique aqui e veja uma ótima referência para estudos de C e TAD.
- Como escrever um arquivo header (`.h`).

5 Como deve ser feita a entrega

A entrega DEVE ser feita pelo Moodle na forma de um único arquivo zipado, contendo o código, os arquivos e a documentação. Também deve ser entregue a documentação impressa (em frente/verso) na próxima aula após a data de entrega do trabalho.

6 Comentários gerais:

- Comece a fazer este trabalho logo, enquanto o problema está fresco na memória e o prazo para terminá-lo está tão longe quanto jamais poderá estar;
- A qualidade da solução/código também faz parte da avaliação;
- Clareza, indentação e comentários no programa também vão valer pontos;
- O trabalho é individual (grupo de UM aluno);
- Trabalhos copiados (documentação e/ou código fonte) terão nota zero;
- Trabalhos entregue em atraso serão aceitos, todavia a nota atribuída ao trabalho será zero;
- Códigos que não compilares serão penalizados.
- Evite discussões inócuas com o professor em tentar postergar a data de entrega do referido trabalho.

Créditos ao prof. Bruno Cerqueira Hott.