Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Biológicas – ICEB
Departamento de Computação – DECOM
Disciplina: Algoritmos e Estrutura de Dados I – CIC102 / 20091
Professor: David Menotti (menottid@gmail.com)

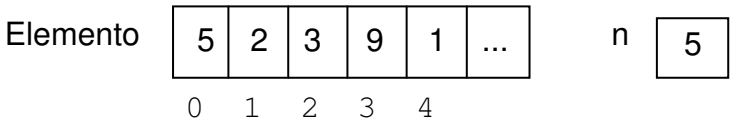
Trabalho Prático 1

Revisão de Programação, Tipos Abstratos de Dados e Combinação de Alocação Estática e Dinâmica de Memória

Valor: 0,5 pontos (5% da nota total) **Data de entrega: 31/03/2009**
Documentação em Latex: +0,1 pontos

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.

Você deverá implementar um tipo abstrato de dados **`TConj`** para representar conjuntos de elemento 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:



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 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 se não.
   `int InserirElemento(TConj* pA, int elem);`
3. Excluir um elemento do conjunto, retornando 1 se realizado e 0 se não.
   `int ExcluirElemento(TConj* pA, int elem);`
4. *Setar* um elemento em uma posição do conjunto, retornando em 1 caso de sucesso e 0 em caso contrário.
   `int SetElemento(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.
   `int GetElemento (TConj A, int pos, int* pelem);`
6. Testar se um elemento está presente no conjunto, retornando sua posição ou -1 se não estiver presente
   `int TestarElemento (TConj A, int elem);`
7. Gerar um conjunto aleatoriamente com n elementos:
   `TConj GerarConjunto(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 conjunto 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 utilizá-lo em **dois programas diferentes**.

O primeiro é o programa de testes mostrado a seguir:

```
main() {
   TConj a,b,c,d;
   int elemento, num;

   Inicializar(a);

   srand(time(NULL));
   for(i=0;i<20;i++)
      InserirElemento(a,rand()%40);
   Imprimir(a);

   srand(time(NULL));
   for(i=0;i<10;i++)
      RemoverElemento(a,rand()%20-1);
   Imprimir(a);

   if (TestarElemento(a,9) == -1)
      printf("o elemento 9 não esta presente em a");
   elemento = GetElemento(a,1);
   b = GerarConjunto(10);
```

```
   num = Conj2Num(b);
   c = Num2Conj(num);
   if (Comparar(b,c))   printf("os conjunto sao iguais\n");
   else                 printf("os conjunto sao diferentes\n");

   d=Unir(a,b);
   Imprimir(d);
   d=Inter(a,b);
   Imprimir(d);
   d=Subtrair(a,b);
   Imprimir(d);
}
```

O segundo programa no qual seu TAD deve ser usado é na implementação do **Jogo da Senha** (por exemplo: http://www.usinadejogos.com.br/senha.html). O jogo da senha gera um número aleatório e deve permitir que o usuário adivinhe este número. Considere que os elementos possíveis para este programa são os algarismos 0, 1, 2, 3, ... 9. A cada tentativa do usuário, o jogo deverá lhe informar quantos elementos estão certos na posição certa e quantos estão certos mas na posição errada Por exemplo:

O jogo gera o número: **`5219.`** E o usuário joga:

Tentativa	Número	Resposta
1	2381	0 2
2	4298	1 1
3	8267	1 0
4	9215	2 2
5	1295	1 3
6	5219	4 0 - Você acertou em 6 tentativas.

O jogo deve permitir 4 níveis de dificuldade (*fácil, difícil* e *teste*) – o nível *fácil* usa senhas de apenas 3 dígitos sem repetição, o nível *difícil* usa senhas de 4 dígitos sem repetição. O modo *teste* é igual ao *difícil*, mas a senha aparece para o usuário. O número de tentativas máximo é um parâmetro a ser solicitado do usuário. A interface do jogo pode ser completamente textual, mas você deve utilizar o TAD criado para implementar o jogo. (**Note que o seu programa principal não poderá acessar diretamente a estrutura interna do TAD. Se necessário, acrescente novas funções ao seu TAD detalhando-as na documentação do trabalho**)

Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Biológicas – ICEB
Departamento de Computação – DECOM
Disciplina: Algoritmos e Estrutura de Dados I – CIC102 / 20091
Professor: David Menotti (menottid@gmail.com)

O que deve ser entregue

- Código fonte do programa em C ou C++ (bem identada e comentada).
- Documentação do trabalho. 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. Em Latex: Caso o trabalho seja elaborado/escrito em latex, ganha-se 0,1 pontos.
  8. Formato: mandatoriamente em PDF (http://www.pdf995.com/).

Obs1: Consulte as dicas do Prof. Nívio Ziviani de como deve ser feita uma boa implementação e documentação de um trabalho prático: http://www.dcc.ufmg.br/~nivio/cursos/aed2/roteiro/

Obs2: Veja modelo de como fazer o trabalho em latex:
http://www.decom.ufop.br/prof/menotti/aedI091/tps/modelo.zip
(caso alguém desenvolva um modelo similar em word, favor me enviar)

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 na próxima aula (teórica ou prática) após a data de entrega do trabalho.

Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Biológicas – ICEB
Departamento de Computação – DECOM
Disciplina: Algoritmos e Estrutura de Dados I – CIC102 / 20091
Professor: David Menotti (menottid@gmail.com)

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;
- Clareza, identação e comentários no programa também vão valer pontos;
- O trabalho é individual (grupo de UM aluno);
- Trabalhos copiados (e FONTE) terão nota zero;
- Trabalhos entregue em atraso serão aceitos, todavia a nota atribuída ao trabalho será zero;
- Evite discussões inócuas com o professor em tentar postergar a data de entrega do referido trabalho.