Universidade Federal do Paraná – UFPR
Setor de Ciências Exatas / Departamento de Informática – DInf
Disciplina: Algoritmos e Estruturas de Dados II – CI056
Professor: David Menotti (menottid@gmail.com)

Lista de Exercícios sobre
Tipos Abstratos de Dados (TAD) e Estruturas de Dados em C

1) Escreva uma especificação de tipos abstratos de dados (TAD) para os números complexos, $a + bi$, onde $abs(a + bi)$ é $sqrt(a^2 + b^2)$, $(a + bi) + (c + di)$ é $(a + c) + (b + d)i$, $(a + b) * (c + di)$ é $(a * c – b * d) + (a * d + b * c)i$ e $–(a + bi)$ é $(-a) + (-b)i$. Então, implemente (cm C/C++ ou Java) números complexos, conforme especificado acima, usando estruturas com partes reais e complexas. Escreva rotinas para somar, multiplicar e negar (inverter) tais números.

2) Vamos supor que um número real seja representado por uma estrutura em C, como esta:

```
struct realtype
{
      int left;
      int right;
};
```

onde *left* e *right* representam os dígitos posicionados à esquerda e à direita do ponto decimal, respectivamnete. Se *left* for um inteiro negativo, o número real representado será negativo.

   a. Escreva uma rotina para inserir um número real e criar uma estrutura representado esse número;
   b. Escreva uma função que aceite essa estrutura e retorne o número real representado por ela.
   c. Escreva rotinas add, substract e multiply que aceitem duas dessas estruturas e definam o valor de uma terceira estrutura para representar o número que seja a soma, a diferença e o produto, respectivamente, dos dois registros de entrada.

3) Suponha que um inteiro precise de quatro bytes, um número real precise de oito bytes e um caractere precise de um byte. Pressuponha as seguintes definições e declarações:

```
struct nametype
{
      char first[10];
      char midinit;
      char last[20];
};
```

```
struct person
{
      struct nametype name;
      int             birthday[2];
      struct nametype parentes[2];
      int             income;
      int             numchildren;
      char            address[20];
      char            city[10];
      char            state[2];
};

struct person p[100];
```

se o endereço inicial de p for 100, quais serão os endereços inicias (em bytes) de cada um dos seguintes?

a. `p[10]`
b. `p[200].name.midinit`
c. `p[20].income`
d. `p[20].address[5]`
e. `p[5].parents[1].last[10]`

4) Suponha dois vetores, um de registros de estudantes e outro de registros de funcionários. Cada registro de estudante contém membros para um último nome, um primeiro nome e um índice de pontos de graduação. Cada registro de funcionário contém membros para um último nome, um primeiro nome e um salário. Ambos os vetores são classificados em ordem alfabética pelo último e pelo primeiro nome. Dois registros com o último e o primeiro nome iguais não aparecem no mesmo vetor. Escreva uma função em C para conceder uma aumento de 10% a todo funcionário que tenha um registro de estudante cujo índice de pontos de graduação seja maior que 3.0.

5) Escreva uma função semelhante à do exercício anterior, mas pressupondo que os registros dos funcionários e estudantes sejam mantidos em dois arquivos externos classificados, em vez de em dois vetores classificados.

6) Usando a representação de números racionais apresentada abaixo, escreva rotinas para somar, subtrair e dividir tais números.

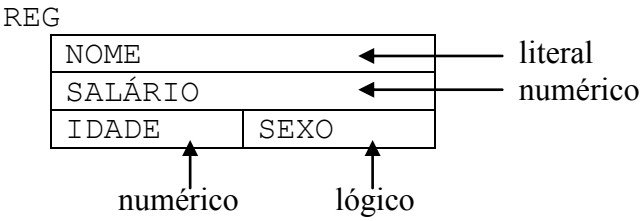
```
typedef struct
{
      int numerator;
      int denominator;
} RATIONAL;
```

```
struct RATIONAL
{
      int numerator;
      int denominator;
};
```

7) Implemente uma função *equal* que determine se dois números racionais, *r*1 e *r*2, são iguais ou não reduzindo primeiramente *r*1 e *r*2 a seus termos mínimos e verificando em seguida a igualdade.

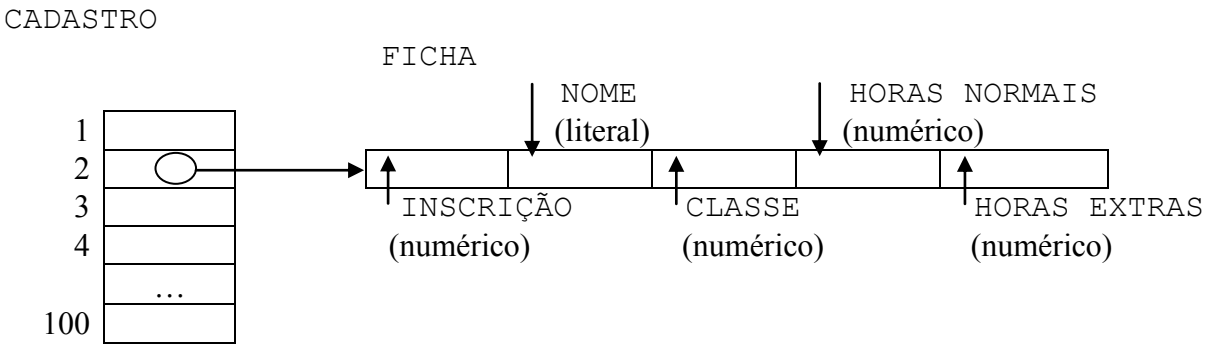
8) Um método alternativo para implementar a função *equal* citada na questão anterior seria multiplicar o denominador de cada número pelo numerador do outro e testar a igualdade dos dois produtos. Escreva uma função *equal2* para implementar esse algoritmo. Qual dos dois métodos é preferível?

9) Definir e declarar o registro cuja representação gráfica é dada a seguir.



10) Escrever uma função para atribuir um valor ao campo de nome SALÁRIO do registro REG, descrito no exercício anterior. Escreva também um programa (função *main*) para utilizar a função criada.

11) Uma indústria faz a folha mensal de pagamentos de seus empregados baseada no seguinte: Existe uma tabela com os dados do funcionário



Fazer um algoritmo que processe a tabela e emita, para cada funcionário seu contracheque cujo formato é dado a seguir:

```
NÚMERO DE INSCRIÇÃO:      NOME:
SALÁRIO HORAS NORMAIS:
SALÁRIO HORAS EXTRAS:
DEDUÇÃO INSS:
  SALÁRIO LÍQUIDO:
```

O salário de referência deverá ser lido previamente.

O salário referente às horas extras é calculado acrescentando 30% ao salário-hora normal.

O desconto do INSS é de 11% do salário bruto (salário correspondente às horas normais trabalhadas + salário correspondente às horas extras).

Para o cálculo do salário, considerar que existem duas classes de funcionários, a classe 1, cujo salário é 1,3 vezes o salário de referência, e a classe 2, cujo salário é 1,9 vezes o salário de referência.

12) Para evitar erros de digitação de sequências de números de importância fundamental, como a matrícula de um aluno, o CPF do Imposto de Renda, o número de conta bancária, geralmente adiciona-se ao número um dígito verificador. Por exemplo, o número de matrícula 811057 é usado como 8110573, onde 3 é o dígito verificador, calculado da seguinte maneira:
    a. Cada algarismo do número e multiplicado por um peso começando de 2 e crescendo de 1 em 1, da direita para a esquerda:
    $$8 \times 7, 1 \times 6, 1 \times 5, 0 \times 4, 5 \times 3, 7 \times 2;$$
    b. Somam-se as parcelas obtidas:
    $$56 + 6 + 5 + 0 + 15 + 14 = 96;$$
    c. Calcula-se o resto da divisão desta soma por 11:
    96 divido por 11 dá resto 8 ($96 = 8 \times 11 + 8$);
    d. Subtrai-se de 11 o resto obtido:
    $$11 - 8 = 3;$$
    e. Se o valor encontrado for 10 ou 11, o dígito verificador será 0; nos outros casos, o dígito verificar é o próprio resto da divisão.

    Escrever um algoritmo capaz de:
    1) Ler um conjunto de registros contendo, cada um, o número de uma conta bancária, o dígito verificador deste número, o saldo da conta e o nome do cliente. O último registro, que não deve ser considerado contém o número de conta igual a zero.
    2) Utilizando o esquema de verificação acima, imprimir duas listas de clientes distintas no seguinte formato de saída:

```
CONTAS DE NÚMERO CORRETO
413599-7    987,30  Jacinto Pereira
111118-     121,99  Campos Sales
06
...

CONTAS DE NÚMERO ERRADO
765432-1    335,66  Júnia Faria
...
```

13) Defina um Tipo Abstrato de Dados `TMatriz`, para representar matrizes quadradas de tamanho n. Implemente as operações para somar e multiplicar 2 matrizes. Implemente ainda a operação do cálculo da matriz inversa.

14) Você deverá implementar um tipo abstrato de dados `TConjunto` para representar conjuntos de números inteiros. Seu tipo abstrato deverá armazenar os elementos do

conjunto e o seu tamanho *n*. Considere que o tamanho máximo de um conjunto é 20 elementos e use arranjos de 1 dimensão (vetores) para a sua implementação. Seu TAD deve possuir procedimentos (ou funções quando for o caso) para:

a. criar um conjunto vazio;
b. ler os dados de um conjunto;
c. fazer a união de dois conjuntos;
d. fazer a interseção de dois conjuntos;
e. verificar se dois conjunto são iguais (possuem os mesmos elementos);
f. imprimir um conjunto;

Exercícios extraídos de (Referências)

[1] Aaron M. Tenenbaum, Yedidyah Langsam, Moshe J. Augenstein, *Estruturas de Dados Usando C*, Makron Books/Pearson Education, 1995.
[2] Aaron M. Tenenbaum, Yedidyah Langsam, Moshe J. Augenstein, *Data Strcutures Using C*, Prentice-Hall International Editions, 1995.
[3] Harry Farrer, Christiano Gonçalves Becker, Eduardo Chaves Faria, Helton Fábio de Matos, Marcos Augusto dos Santos, Miriam Lourenço Maia, *Algoritmos Estruturados*, LTC Editora, 3ª. edição, Rio de Janeiro, 1999.