Universidade Federal de Ouro Preto - UFOP
Instituto de Ciências Exatas e Biológicas - ICEB
Departamento de Computação - DECOM
Disciplina: BCC-201 Introdução à Programação

Exercícios de Estruturas

1. Escreva uma especificação para os números complexos, $a + bi$, onde $abs(a + bi)$ é $sqrt(a^2 + bi^2)$, $(a+bi)+(c+di)$ é $(a+c)+(b+d)i$. Então, implemente números complexos, conforme especificado acima, usando estruturas com partes reais e complexas. Escreva rotinas para somar e encontrar o valor absoluto de 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, respectivamente. 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. Escreva rotinas para somar, subtrair e dividir números racionais (numerador/denominador).

```
struct Racional
{
    int numerador;
    int denominador;
};
```

4. Implemente uma função equal que determine se dois números racionais, $r1$ e $r2$, são iguais ou não reduzindo primeiramente $r1$ e $r2$ a seus termos mínimos e verificando em seguida a igualdade.

5. Você deverá implementar uma estrutura TConjunto para representar conjuntos de números inteiros. Sua estrutura 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. Implementar as seguintes operações:

   (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;

6. Seja a seguinte estrutura:

```
struct time {
    int hora;
    int min;
    int seg;
};
```

Fazer um programa contendo funções para:

(a) Receber uma quantidade de tempo em segundos e transformar a mesma em horas, minutos e segundo, preenchendo e imprimindo a variável tempo.

(b) Ler a variável tempo (campos hora, min e seg) e devolver a quantidade de horas em segundos.

7. Seja uma estrutura para descrever os carros de uma determinada revendedora, contendo os seguintes campos:

marca: *string* de tamanho 15

ano: inteiro

cor: *string* de tamanho 10

preço: real

(a) Escrever a definição da estrutura carro.

(b) Declarar o vetor vetcarros do tipo da estrutura definida acima, de tamanho 20.

(c) Definir uma função para ler o vetor vetcarros.

(d) Definir uma função que receba um preço e imprima os carros (marca, cor e ano) que tenham preço igual ou menor ao preço recebido.

(e) Defina uma função que leia a marca de um carro e imprima as informações de todos os carros dessa marca (preço, ano e cor).

(f) Defina uma função que leia uma marca, ano e cor e informe se existe ou não um carro com essas características. Se existir, informar o preço.

8. Seja uma estrutura contendo dados dos funcionários de uma empresa. Para cada funcionário tem-se os seguintes dados:

nome: *string* de tamanho 20

CPF: vetor de 11 inteiros

numpeças: inteiro

salário: real

Seja os seguintes trechos de programa:

```
main( )
{
    struct funcionário vetfunc[50] ;
    int nfunc; // numero de funcionarios
    printf("Digite número de funcionários (<= 50): ");
    scanf("%d", &nfunc);
    leitura (vetfunc, nfunc);
    calc_salario( );
    tot_folha( );
    printf(" Total de peças fabricadas no mês = %d \n", tot_peças( ) );
    op_maior_sal( );
}

void leitura(struct funcionário vetfunc[], int nfunc)
{
    int i,j;
    for (i = 0; i < nfunc; i++)
    {
        le_CPF(vetfunc[i]);
        while (ver_CPF(vetfunc[i].CPF) == 0)
        {
            printf(" \n Erro na digitação do CPF. \n"):
            le_CPF(vetfunc[i]);
        }
        le_outros_dados(vetfunc[i]);
    }
}
```

Escrever a definição das funções que estão faltando.

Observações:

Obs.1: A função `leitura()` chama três funções:

le CPF(): executa a leitura dos 11 dígitos que compõe o CPF

ver CPF(): verifica se o CPF é valido, se for retorna o valor 1, caso contrário retorna 0.

le outros dados(): Executa a leitura dos campos `nome` e `num_peças`.

Para verificar a validade do CPF (ou CIC):

Um número de CPF é seguido de dois dígitos denominados dígitos de controle. Estes dígitos são gerados a partir dos dígitos que compõem o número de um CPF e acompanham este número como sufixo. Digitando-se um número de CPF é possível computar os seus dígitos de controle e compará-los com os fornecidos ao sistema. Se os dígitos computados não batem com os dígitos fornecidos, então o número do CPF é falso ou ocorreu um erro de digitação. Se eles batem, então as chances de que o número seja correto são muito altas.

Seja $CPF = x[0]x[1]...x[8]x[9]x[10]$, onde $x[i]$ representa um dígito do CPF para $0 \leq i \leq 8$ e $x[i]$ um dígito de controle para $9 \leq i \leq 10$.

Exemplo:

Seja o CPF: 1 0 3 1 2 4 9 2 1 $X[9]$ $X[10]$.

O dígito de controle $X[9]$ é obtido da seguinte maneira:

- multiplicar os dígitos da esquerda para a direita por um número começando de 1 e incrementado de 1 (de X[0] até x[8]):

  Ex.: 1*1, 0*2, 3*3, 1*4, 2*5, 4*6, 9*7, 2*8, 1*9

- Somam-se as parcelas obtidas:

  Ex.: 1 + 0 + 9 + 4 + 10 + 24 + 63 + 16 + 9 = 136

  Obtem-se o resto da divisão inteira desta soma por 11: 138%11 = 4, 4 corresponde ao dígito $X[9]$.

O dígito de controle $X[10]$ é obtido da seguinte maneira:

- multiplicar os dígitos da esquerda para a direita por um número começando de 9 e decrementado de 1 (de $X[0]$ até $x[8]$):

  Ex.: 1*9, 0*8, 3*7, 1*6, 2*5, 4*4, 9*3, 2*2, 1*1

  Somam-se as parcelas obtidas:

  Ex.: 9 + 0 + 21 + 6 + 10 + 16 + 27 + 4 + 1 = 94

  Obtem-se o resto da divisão inteira desta soma por 11: 94%11 = 6, 6 corresponde ao dígito $X[10]$. Se o resto da divisão for igual a 10, deve-se considerar como 0. `Ex.: 98%11 = 10` = 0.
  A função deve verificar se os dígitos de controle fornecidos pelo usuário, obedecem a regra acima.

Obs.2: A função `calc_salario()` é responsável por calcular o campo salário de cada funcionário, através da seguinte regra que considera o número de peças fabricadas:

- Para número de peças menor ou igual a 30: recebe salário mínimo (`R$230,00`)
- Para número de peças entre 31 e 45: salário mínimo mais 3% do salário por peça acima das iniciais.
- Para número de peças acima de 45: salário mínimo mais 5% do salário por peça acima das 30 iniciais.

Obs.3: A função `tot_folha( )` calcula e imprime o valor da folha de pagamento da empresa.

Obs.4: A função `tot_peças( )` retorna o número total de peças fabricadas na empresa.

Obs.5: A função `op_maior_sal( )` imprime os dados do operário de maior salário.