UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

PROCESSO SELETIVO – 005 / 2022

Área de Conhecimento: Ciência da Computação / Algoritmos e Estrutura de Dados

PROVA ESCRITA – PADRÃO DE RESPOSTA

QUESTÃO 1: Utilizando a Linguagem C, crie uma estrutura de dados (***struct***) para armazenar dados de um livro. Ela deve possuir campos para título, autor, código e preço. Faça um programa para ler inicialmente a quantidade de livros e depois ler os dados da entrada padrão e armazenar esses dados em um vetor. A seguir, o programa deve mostrar na tela os títulos dos livros com preço menor ou igual a um valor informado pelo usuário.

```
#include <stdio.h>
#include <stdlib.h>

struct livro{
  char titulo[30];
  char autor[30];
  int codigo;
  float preco;
};

int main(int argc, char *argv[]) {
  int n, i;
  printf("Informe a quantidade n de livros: ");
  scanf("%d", &n);
  struct livro v[n];
  // struct livro *v = malloc( sizeof(struct livro) * n ); // outra forma.
  for( i = 0 ; i < n ; i++ ){
    printf("Entrada de dados do livro %d:\n", i+1 );
    printf("Digite o titulo: ");
    scanf("%s", v[i].titulo);
    printf("Digite o autor: ");
    scanf("%s", v[i].autor);
    printf("Digite o codigo: ");
    scanf("%d", &v[i].codigo);
    printf("Digite o preco: ");
    scanf("%f", &v[i].preco);
  }
  float valor;
  printf("Digite um valor: ");
  scanf("%f", &valor);
  printf("Livros com preco menor ou igual a R$%.2f:\n", valor);
  for( i = 0 ; i < n ; i++ )
    if( v[i].preco <= valor )
      printf("%s\n", v[i].titulo);
  //free( v ); // caso tenha usado o malloc().
  return 0;
}
```

Membros da Banca:

Gilmario Barbosa dos Santos	**Janine Kniess**
Yuri Kaszubowski Lopes	**Rui Jorge Tramontin Jr. (Presidente)**

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

PROCESSO SELETIVO – 005 / 2022

Área de Conhecimento: Ciência da Computação / Algoritmos e Estrutura de Dados

PROVA ESCRITA – PADRÃO DE RESPOSTA

QUESTÃO 2: Arquivos são um mecanismo de abstração que permitem a manipulação de dados de maneira persistente, concorrente e em grandes quantidades. Sobre o assunto, discorra sobre as formas de abertura, leitura, escrita e fechamento de arquivos na linguagem C. Para complementar a explicação, apresente trechos de código demonstrando as formas de manipulação citadas.

As funções de manipulação de arquivos são oferecidas pela biblioteca **stdio.h**. Para se ter acesso a um arquivo, é preciso realizar a sua abertura, processo que é feito via chamada da função *fopen()*, que tem o seguinte protótipo:

```
FILE *fopen( char *nome_arq, char *modo );
```

- o parâmetro *nome_arq* é contém o nome do arquivo a ser aberto;
- o parâmetro *modo* define como o arquivo vai ser aberto (leitura, escrita, etc.) e se será em modo de **texto** ou **binário**.

Exemplos de abertura ilustrando alguns modos:

```
FILE *f = fopen( "entrada.txt", "rb" ); // Abre para leitura em modo texto
FILE *f = fopen( "log.txt", "at" ); // Abre para fazer append em modo texto
```

- O retorno da função é um ponteiro do tipo FILE. Tal estrutura permite que o programa possa ter acesso ao arquivo e deve ser passada como parâmetro para as funções de manipulação. Caso o arquivo não possa ser aberto, a função retorna um ponteiro NULL. Alguns fatores que levam a erro na abertura são : (i) arquivo inexistente (no caso de leitura); (ii) espaço insuficiente em disco (para gravação); (iii) S.O. não dá permissão de acesso ao arquivo.

Após o uso, o arquivo deve ser fechado. O fechamento é fundamental, sobretudo após operações de gravação. Para fechar um aquivo, basta chamar a função *fclose()*, passando o descritor do arquivo (ponteiro FILE) como parâmetro.

Quando aberto em **modo texto**, um arquivo é interpretado como sequências de caracteres agrupadas em linhas. A mudança de linha é feita pelo caractere *line feed* ('\r'). A manipulação de arquivos em modo texto pode ser feita das seguintes formas:

- Entrada/saída formatada: funções *fscanf()* e *fprintf()*. São variações das funções já conhecidas para E/S via *console*;
- Entrada/saída linha a linha (salvas em *strings*): Funções *fgets()* e *puts()*;
- Entrada/saída por caractere individual (tipo *char*): funções *getc()* e *putc()*.

Quando aberto em **modo binário**, o arquivo contém blocos da dados, cujo formato depende do seu conteúdo. São utilizadas as funções *fread()* para leitura e *fwrite()* para escrita de blocos de dados. Por exemplo, assumindo um descritor de arquivo identificado por **f**, o vetor **v** de livros da questão 1 pode ser armazenado da seguinte forma:

```
fwrite( v, sizeof( struct livro ), n, f );
```

Membros da Banca:

Gilmario Barbosa dos Santos	**Janine Kniess**
Yuri Kaszubowski Lopes	**Rui Jorge Tramontin Jr. (Presidente)**

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

PROCESSO SELETIVO – 005 / 2022

Área de Conhecimento: Ciência da Computação / Algoritmos e Estrutura de Dados

PROVA ESCRITA – PADRÃO DE RESPOSTA

QUESTÃO 3: Existem duas formas de passagem de parâmetro em funções: por valor e por referência. Na linguagem C, todas as chamadas são feitas por valor, mas é possível simular a chamada por referência usando operadores de endereço e de indireção. Considerando essas duas formas de passagem de parâmetros, explique como deve ser feito para cada situação (por exemplo: tipo primitivo, estrutura definida pelo usuário, matriz, entre outros). Exemplifique tais situações utilizando trechos de código na linguagem C.

O mecanismo de passagem de parâmetros permite com que o código de funções possa ser reutilizado, facilitando a depuração e escalabilidade de um programa. Na passagem de parâmetro **por valor**, uma cópia do valor da variável é passada ao parâmetro declarado na função. No exemplo a seguir, temos uma função que calcula a soma entre x e y:

```
int soma( int x, int y ){
  return x + y;
}
```

No programa principal, a chamada da função ficaria assim, por exemplo:

```
...
int a, b;
// Entrada de dados de a e b...
int result = soma( a, b );
...
```

Como a passagem é por valor, um valor literal poderia ser passado como parâmetro. Por exemplo, para somar a + 1, poder-se-ia fazer: `int result = soma( a, 1 );`

A limitação desse tipo de passagem é que a variável não pode ser modificada pela função, pois somente o seu valor é passado. Nas situações em que é preciso modificar uma variável fora do escopo da função, usa-se uma **passagem por referência**. Nesse tipo de passagem, a função deve receber o endereço da variável (sua referência), utilizando o operador **&**. O parâmetro da função deve ser, portanto, um ponteiro, que vai fazer o acesso à variável utilizando o operador de indireção (*****). No exemplo a seguir temos uma função que incrementa o valor de uma variável:

```
void inc( int *p ){
  (*p)++; // O operador * permite o acesso à variável apontada por p.
}
```

A chamada da função é feita da seguinte forma:

```
...
int a = 1;
int( &a ); // Passa o endereço de a para a função.
// Depois da chamada, a tem o valor 2.
```

É importante destacar que vetores (e também matrizes, pois em C matrizes são vetores de vetores) sempre são passados por referência. Ou seja, não é necessário utilizar o operador **&** quando um vetor é passado como parâmetro. No exemplo a seguir temos uma função que faz a entrada de dados em um vetor:

```
void entrada( int v[], int n ){
  int i;
  for( i = 0 ; i < n ; i++ )
    scanf("%f", &v[i] );
}
```

Repare que o parâmetro **v** é, na prática, um ponteiro, podendo também ser declarado como tal. A chamada da função fica assim:

```
...
float vet[10];
entrada( vet, 10 );
// Depois da chamada, o vetor tem os valores conforme definidos dentro da função.
```

Por último, podemos falar dos tipos estruturados (*struct*) que, assim como variáveis de tipo primitivo, a passagem de parâmetro por referência deve ser feita explicitamente utilizando o operador **&**. Nesse caso, o parâmetro que é o ponteiro utiliza o operador **->** para o acesso aos campos da estrutura apontada por ele. No exemplo a seguir, temos uma função que faz a entrada de dados de uma estrutura de tipo *livro*, definida na questão 1:

```
void entrada( struct livro *p ){
  scanf("%s", p->titulo);
  scanf("%s", p->autor);
  scanf("%d", &p->codigo);
  scanf("%f", &p->preco);
}
```

A chamada da função seria da seguinte forma:

```
...
struct livro x;
entrada( &x );
// Após a chamada, os valores definidos na função estão armazenados em x.
```

Membros da Banca:

Gilmario Barbosa dos Santos	**Janine Kniess**
Yuri Kaszubowski Lopes	**Rui Jorge Tramontin Jr. (Presidente)**

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

PROCESSO SELETIVO – 005 / 2022

Área de Conhecimento: Ciência da Computação / Algoritmos e Estrutura de Dados

PROVA ESCRITA – PADRÃO DE RESPOSTA

QUESTÃO 4: Considere o tipo de *nó* de uma *Árvore Binária de Busca* (ABB) definido a seguir:

```
struct nodeType{
 int value;
 struct nodeType *left, *right;
}
```

Escreva uma função recursiva em C que insere um novo valor na árvore. A função recebe como parâmetros um nó e o valor (*int*) a ser inserido.

```
/*
 A função deve ser chamada no programa passando como parâmetros:
  - o ponteiro para o nó raiz da árvore;
  - o valor a ser inserido (info).

 Ainda na chamada, o retorno da função deve ser atribuído ao ponteiro para o nó raiz.
*/

struct nodeType *insert( struct nodeType *n, int info ){

  if( n == NULL ){ // Árvore vazia (ou fim do percurso)
    // Aloca novo nó.
    struct nodeType *p = malloc( sizeof( struct nodeType ) );
    if( p == NULL) // Caso dê erro na alocação
      return NULL;
    p->value = info;
    p->left = p->right = NULL;
    return p;
  }

  if( info < n->value ) // Percore recursivamente à esquerda.
    n->left = insert( n->left, info );
  else
    if( info > n->value ) // Percore recursivamente à direita.
      n->right = insert( n->right, info );

  return n; // Retorna o próprio nó.
}
```

Membros da Banca:

Gilmario Barbosa dos Santos	**Janine Kniess**
Yuri Kaszubowski Lopes	**Rui Jorge Tramontin Jr. (Presidente)**

Assinaturas do documento

Código para verificação: **64SU2Y6G**

Este documento foi assinado digitalmente pelos seguintes signatários nas datas indicadas:

RUI JORGE TRAMONTIN JUNIOR (CPF: 020.XXX.169-XX) em 12/12/2022 às 09:42:22
Emitido por: "SGP-e", emitido em 30/03/2018 - 12:44:55 e válido até 30/03/2118 - 12:44:55.
(Assinatura do sistema)

GILMARIO BARBOSA DOS SANTOS (CPF: 410.XXX.285-XX) em 12/12/2022 às 09:44:15
Emitido por: "SGP-e", emitido em 30/03/2018 - 12:35:07 e válido até 30/03/2118 - 12:35:07.
(Assinatura do sistema)

Para verificar a autenticidade desta cópia, acesse o link https://portal.sgpe.sea.sc.gov.br/portal-externo/conferencia-documento/VURFU0NfMTIwMjJfMDAwNTYwMzFfNTYxMThfMjAyMl82NFNVMlk2Rw== ou o site https://portal.sgpe.sea.sc.gov.br/portal-externo e informe o processo **UDESC 00056031/2022** e o código **64SU2Y6G** ou aponte a câmera para o QR Code presente nesta página para realizar a conferência.