REPÚBLICA PORTUGUESA
EDUCAÇÃO

MATRIZ

Curso Profissional Técnico/a Programador de Informática

Programação e Sistemas Distribuídos

UFCD 0810 - Programação em C/C++ - avançada

Julho de 2020

Modalidade da prova: teórica

Duração: 90 minutos

Objeto de avaliação

A prova tem por referência o programa da UFCD 0810, Programação em C/C++ - avançada, da disciplina de Programação e Sistemas Distribuídos.

Conteúdos

- Apontadores
  - Definição de apontadores
  - Declaração de apontadores
  - Operador indireto (*)
  - Atribuição de valores a variáveis apontadores
  - Operações com apontadores
  - Apontadores para apontadores

- Estruturas
  - Definição e declaração de estruturas
  - Inicialização de estruturas
  - Leitura e escrita de valores nos elementos de uma estrutura
  - Apontadores para estruturas
- Campos bit
  - Diretivas # include e #define

- Livraria do C/C++
    - Acesso às livrarias stdio.h , string.h e stdlib.h

- Ficheiros em C/C++
    - Níveis de leitura e escrita em ficheiros
    - Abertura e fecho de ficheiros
    - Leitura e escrita em ficheiros
    - Outras funções para manipular ficheiros

Objetivos

• Elaborar programas complexos em linguagem C/C++ ;

• Interpretar código, referente a apontadores, escrito em linguagem C/C++.

Estrutura/Cotações

Estrutura	Cotação
2 Questões sobre apontadores	30 a 35 pontos
2 Questões sobre estruturas	80 a 85 pontos
2 Questões sobre ficheiros	75 a 85 pontos
1 Questão sobre funções para manipular ficheiros	10 a 15 pontos
TOTAL:	200 pontos

Material

As respostas são registadas em folha própria, fornecida pelo estabelecimento de ensino (modelo oficial). Como material de escrita, apenas pode ser usada caneta ou esferográfica de tinta azul ou preta.

Critérios gerais de classificação

- A cotação de cada questão ou alínea terá sempre um número inteiro de pontos;
- A cada tarefa executada corretamente será atribuída a cotação total; por cada tarefa não realizada a cotação a atribuir será zero;
- A cotação, de tarefas cuja execução seja composta de vários passos, será fracionada de modo a contemplar os conhecimentos revelados quando a resolução não estiver totalmente correta.

ANEXO À PROVA

Formatos, cláusulas, sintaxe de estruturas, comandos e funções de ficheiros e modos de abertura de ficheiros em C

Formatos de leitura e de escrita de dados

Tipo		Formato
caráter	char	%c
Cadeia de carateres	…	%s
inteiro	int (-32.768 a 32.767)	%d
	long int (-2.147.483.648 a 2.147.483.647)	%ld
	unsigned long int (0 a 4.294.967.295)	%lu
	unsigned int (0 a 65.535)	%u
real	Float (3.4 E-38 a 3.4E+38)	%f ou %e ou %E
	Double (3.4E-4932 a 1.1E+4932)	%f ou %e ou %E

Modificadores de Tipo	Descrição
signed	Indica que o valor terá sinal positivo ou negativo.
unsigned	Indica que o valor não terá sinal, ou seja, é sempre positivo.
short	Indica um inteiro pequeno (2 bytes).
long	Indica um inteiro longo (4 bytes).

Sintaxe:

```
switch (expressão)
{
   case  constante1 : instrução1;
   case  constante2 : instrução2;
         …
   case  constanten : instruçãon;
   [default : instruçãox;]
}
```

Sintaxe (ciclo while):

```
while (condição)
   <Instrução/ões>
```

Sintaxe (ciclo do while):

```
do
         <Instrução/ões>
while (condição);
```

Sintaxe (ciclo for):

```
for (inicializações ; condição ; pós-condição)
   <Instrução/ões>
```

Sintaxe (Função strcpy):

```
char *strcpy(char *dest, char *orig)
```

Copia a string orig para a string dest

DECLARAÇÃO DE ESTRUTURAS:

```
struct [nome_da_estrutura]
{
        tipo1   campo1, campo2 ;
        …
        tipon   campo ;
} ;
```

Declaração de variáveis do tipo estrutura aquando a declaração da própria estrutura

Sintaxe:

```
struct [nome_da_estrutura]
{
        tipo1   campo1, campo2 ;
        …
        tipon   campo ;
}  v1, v2, ..., vn;
```

Declaração de variáveis do tipo estrutura fora da definição da estrutura

Sintaxe:

```
struct [nome_estrutura]    campo1, …, campoN ;
```

FICHEIROS:

ABERTURA DE UM FICHEIRO

Sintaxe:

```
FILE * fopen(const char *filename,const char * mode)
```

Filename — string contend p nome físico do ficheiro
Mode — string contendo o modo de abertura do ficheiro
Nota: Os parâmetros são constantes (const), pois não irão ser alterados dentro da função.

Se a função fopen conseguir abrir um ficheiro com sucesso, cria em memória uma estrutura (do tipo FILE) quer representa toda a informação necessária relativa ao ficheiro que estiver a processar, devolvendo o endereço em que essa estrutura foi criada. Caso não tenha conseguido abrir o ficheiro, devolve NULL.

MODOS DE ABERTURA DE UM FICHEIRO

Modo de abertura	Descrição	Leitura	Escrita	Se ficheiro não existe	Se ficheiro já existe	Posição inicial
r	Ler	SIM	Não	NULL	OK	Início
w	Escrever	Não	SIM	Cria	Recria	Início
a	Acrescentar	Não	SIM	Cria	OK	Fim
r+	Ler/Escrever	SIM	SIM	Cria	Altera dados	Início
w+	Ler/Escrever	SIM	SIM	Cria	Recria	Início
a+	Ler/Escrever	SIM	SIM	Cria	Acrescenta dados	Fim

O modo de abertura pode ainda ser combinado com o tipo de processamento que se pretende dar ao ficheiro:
"t"— ficheiro de texto;
"b"— ficheiro binário. Exemplo: "rb", "wa"

FECHO DE UM FICHEIRO

Sintaxe:

int fclose (FILE *fich)

Esta função devolve 0 em caso de sucesso ou a constante EOF em caso de erro.

LEITURA DE CARATERES DE UM FICHEIRO

Existem várias funções para ler carateres de um ficheiro sendo amais utilizada a seguinte:

int fgetc(FILE *fich);

A função fgetc lê um caráter no ficheiro passado como parâmetro (previamente aberto pela função fopen) e devolve-o como resultado da função. Se não existir qualquer caráter no ficheiro, isto é, se a função detetar uma situação de end-of-file devolve a constante EOF.

- A constante simbólica EOF(-1) serve de constante indicadora de end-of-file.
- Todas as funções de leitura se posicionam automaticamente a seguir ao valor lido do ficheiro não tendo o utilizador que movimentar o apontador para a posição seguinte no ficheiro.

ESCRITA DE CARATERES DE UM FICHEIRO DE TEXTO

A escrita de carateres de um ficheiro pode ser realizada recorrendo à função:

int fputc (int ch, FILE *fich);
que escreve o caráter *ch* no ficheiro *fich*. Esta função devolve o carater *ch* em caso de sucesso ou EOF, caso contrário.

Função fgets

char * fgets (char * str, int num, FILE * stream);

A função fgets é a variante do gets para ficheiros. O fgets tem 3 argumentos:

- um array de carateres onde a linha vai ser guardada;
- um número que indica o número máximo de carateres que pode ser lido (incluindo o '\0').
- o ficheiro de onde pretendemos ler.

Se chegarmos ao fim do ficheiro ou acontecer outro erro qualquer, o fgets retorna NULL, caso contrário a função devolve a string lida e apontada por str.

Função fputs

int *fputs* (const char * str, FILE * stream);

que devolve um valor não negative caso a escrita seja realizada com sucesso ou EOF caso contrário.

INPUT E OUTPUT FORMATADO COM FICHEIROS DE TEXTO

Para o I/O formatado utilizam-se as funções fscanf e fprintf, que funcionam da mesma forma que as funções scanf e printf, tendo apenas mais um parâmetro inicial que corresponde ao ficheiro onde o processamento irá ser realizado.

int fscanf(FILE *fich, const char *format,…)

int fprintf(FILE *fich, const char *format,…)

- A função fscanf devolve EOF se tiver detetado end-of-file ou devolve o n.º de parâmetros que conseguiu ler com sucesso.

FICHEIROS binários

As funções de leitura e escrita que permitem acesso direto são fread e fwrite

fwrite

Faz parte da biblioteca stdio.h e é responsável por escrever um bloco de bytes existentes em memória para um ficheiro aberto em modo binário. Devolve 0 nº de itens que se conseguirem escrever com sucesso.

Sintaxe:

```
int fwrite (const void *ptr, int size, int n, FILE *fich)
```

em que:

ptr – é um apontador para void (isto é para qualquer tipo) e contém o endereço de memória daquilo que queremos guardar em ficheiro. const indica que o parâmetro não será alterado.

size – indica o tamanho em bytes de cada um dos elementos que pretendemos escrever

n – indica o número de elementos que queremos escrever

fich – indica o ficheiro onde serão colocados os dados, este argumento é a variável que recebeu o resultado da função *fopen*

FUNÇÕES DE MANIPULAÇÃO DE FICHEIROS

Posicionar o ponteiro no início do ficheiro (voltar ao início) usa-se a função rewind com a sintaxe seguinte:

Sintaxe:

```
rewind(FILE *fich)
```

fich – Ficheiro sobre o qual se pretende operar.

- Mover o ponteiro do ficheiro para uma posição preestabelecida a usar em ficheiros previamente abertos.

Sintaxe:

```
int fseek (FILE *fich,long salto, int origem)
```

fich – Ficheiro sobre o qual se pretende operar.

salto – (ou offset) indica o nº de bytes que se pretende "andar" (um valor positivo indica que se pretende avançar, um valor negativo indica que se pretende recuar).

origem – Indica o local a partir do qual se quer realizar o salto no ficheiro:

Constante	Valor	Significado
SEEK_SET	0	O salto é realizado a partir da origem do ficheiro
SEEK_CUR	1	O salto é realizado a partir da posição corrente do ficheiro
SEEK_END	2	O salto é realizado a partir do final do ficheiro

A função seek devolve 0 se o movimento dentro do ficheiro realizado com sucesso. Caso contrário devolve um valor diferente de zero.