ARQUIVOS	ARQUIVOS
Qual a importância do uso de arquivos no nossos sistemas?	Já vimos como podemos receber e enviar dados para usuário por meio do teclado e da tela; agora veremos também como ler e gravar dados em arquivos, o que é aliás muito importante ou até essencial em muitas aplicações. A principal vantagem na utilização de arquivo está no fato de as informações ficarem armazenadas num meio físico e não somente em memória como visto até agora.
De que 2 formas podemos armazenar arquivos por através dos nossos programas?	Um arquivo pode ser visto de duas maneiras, na maioria dos sistemas operacionais: em **"modo texto"**, como um texto composto de uma sequência de caracteres, ou em **"modo binário"**, como uma sequência de bytes (números binários). Podemos optar por salvar (e recuperar) informações em disco em um dos dois modos, texto ou binário. Uma vantagem do arquivo texto é que pode ser lido por uma pessoa e editado com editores de textos convencionais. Em contrapartida, com o uso de um arquivo binário é possível salvar (e recuperar) grandes quantidades de informação de forma mais eficiente.
Que operações básicas podemos fazer com os nossos arquivos?	Na manipulação de um arquivo, há basicamente três etapas que precisam ser realizadas: **- abrir o arquivo;** **- ler e/ou gravar os dados desejados;** **- fechar o arquivo;**
O que podemos gravar dados dentro de arquivos por através da Linguagem C?	Assim como as funções de entrada/saída padrão (teclado e tela), as funções de entrada/saída em arquivos estão declaradas no cabeçalho stdio.h. Aliás, as funções para manipulação de arquivos são muito semelhantes às usadas para entrada/saída padrão. Como já dissemos na seção sobre a entrada e saída padrões, a manipulação de arquivos também se dá por meio de fluxos (streams).
Como podemos declarar um determinado dado como arquivo e utilizá-lo na Linguagem C?	A manipulação de um arquivo em linguagem c ocorre com a definição do tipo FILE que tem como objetivo fazer a comunicação entre a memória principal (RAM) e memória secundária (meios magnéticos), por meio do programa e do sistema operacional. Em C, todas as operações realizadas com arquivos envolvem seu identificador de fluxo, que é uma variável do tipo "ponteiro de arquivo" (FILE *). Para declarar um identificador de fluxo, faça como se fosse uma variável normal, de acordo com a codificação a seguir.
- reservada para a questão acima -	***FILE *pont_arq; // não se esqueça do asterisco! O tipo FILE deve ser escrito em maiúsculo.*** "pont_arq" será então um ponteiro para um arquivo. É usando este tipo de ponteiro que vamos poder manipular arquivos na linguagem C.
Quais são as funções mais comuns que podemos utilizar para arquivos no **"modo texto"**?	**Função** **Operação** fopen() Abre um fluxo fclose() Fecha um fluxo putc() Escreve um caractere para um fluxo getc() Lê um caractere para um fluxo fseek() Procura por um byte especificado no fluxo fprintf() É para um fluxo aquilo que printf() é para o console fscanf() É para um fluxo aquilo que scanf() é para o console feof() Retorna verdadeiro se o fim do arquivo é encontrado ferror() Retorna verdadeiro se ocorreu um erro fread() Lê um bloco de dados de um fluxo fwrite() Escreve um bloco de dados para um fluxo rewind() Reposiciona o localizador de posição de arquivo no começo do arquivo remove() Apaga um arquivo
- reservada para a questão acima -	
- reservada para a questão acima -	
Como podemos usar a função fopen para abrir arquivos?	Podemos usar a seguinte sintaxe: ***FILE *pont_arq;*** ***pont_arq = fopen ("exemplo.txt","w");*** Perceba que 1º determinamos o nome do arquivo, caso o arquivo não esteja no mesmo diretório, temos que mostrar o caminho até o diretório. E como segundo argumento da função fopen dizemos qual será o método de acesso, que poderá ser 3: "r" (somente leitura), "a" (para acrescentar dados a um arquivo já existente) e "w" (para sobrescrever um arquivo já exitente ou então criar um novo);

ARQUIVOS	ARQUIVOS
Qual a relevância da função "exit" na manipulação de arquivos de texto?	Ela tem o papel de parar uma execução caso algo dê errado, por exemplo, tentamos abrir um arquivo, mas por algum motivo o arquivo não existe mais ou está corrompido. Podemos gerar um laço condicional que faça o programa terminar caso o arquivo não esteja apto para ser aberto. Por exemplo: ***if (pont_arq==NULL)*** ***{*** ***printf ("Erro na abertura do arquivo.");*** ***exit 1;*** ***}***
Como podemos usar a função fclose para fechar arquivos?	Ao terminar de usar um arquivo, você deve fechá-lo. A função fclose() é usada para fechar um fluxo que foi aberto por uma chamada à função fopen(). Sua sintaxe é: ***int fclose (FILE *pont_arq);*** É importante que se perceba que se deve tomar o maior cuidado para não se "perder" o ponteiro do arquivo. "Perder" neste caso seria se atribuir um valor de outro ponteiro qualquer ao ponteiro de arquivo (perdendo assim o valor original). É utilizando este ponteiro que vamos poder trabalhar com o arquivo. Se perdermos o ponteiro de um determinado arquivo não poderemos nem fechá-lo.
Como podemos apresentar dados de um arquivo do tipo texto por através das funções?	Há quatro funções, das quais três são análogas às usadas para saída padrão: fputc, fputs e fprintf. A tabela a seguir mostra cada função e suas utilidades. **Função Explicação** fputc Imprime apenas um caractere. fputs Imprime uma string diretamente, sem nenhuma formatação. fprintf Imprime uma string formatada. fwrite Grava dados binários para um arquivo. Podemos utilizá-las das seguintes formas: ***void fputc (int caractere, FILE *fluxo);*** ***void fputs (char *string, FILE *fluxo);*** ***void fprintf (FILE *fluxo, char *formatação, ...);*** ***int fwrite (void *dados, int tamanho_do_elemento, int num_elementos, FILE *fluxo);***
- reservada para a questão acima -	
- reservada para a questão acima -	
Como podemos ler dados de um arquivo do tipo texto por através das funções?	Novamente, há quatro funções, das quais três se assemelham às usadas para a saída padrão: **- fgetc, fgets:** ao chamar a função fgets(), você deve fornecer o ponteiro para a string onde os dados lidos devem ser guardados, além do tamanho máximo dos dados a serem lidos (para que a memória reservada à string não seja ultrapassada). **- fscanf:** sintaxe quase igual à de scanf(); só é necessário adicionar o identificador de fluxo no início. A tabela a seguir mostra cada função e suas utilidades. **Função Explicação** fgetc Recebe apenas um caractere. fgets Lê uma string (geralmente uma linha inteira). fscanf Recebe uma string formatada. fread Lê dados binários de um arquivo. A seguir apresentamos os protótipos dessas funções: ***int fgetc (FILE *fluxo)*** ***void fgets (char *string, int tamanho, FILE *fluxo)*** ***void fscanf (FILE *fluxo, char *formatação, ...)*** ***int fread (void *dados, int tamanho_do_elemento, int num_elementos, FILE *fluxo)***
- reservada para a questão acima -	
- reservada para a questão acima -	
Como podemos operar sobre arquivos no Modo Binário?	As funções para abertura e fechamento de um arquivo binário são as mesmas utilizadas em um arquivo texto (fopen e fclose). Entretanto, devemos utilizar o caractere b para o modo de abertura do arquivo. No trecho de código abaixo, é possível observar a abertura do arquivo "pontos.dat" e a utilização da função exit no caso do arquivo não existir ou estiver corrompido. ***FILE *pont_arq;*** ***pont_arq = fopen ("clientes.dat","wb");*** ***if (pont_arq == NULL)*** ***{*** ***printf ("Erro na abertura do arquivo.");*** ***exit 1;*** ***}***
- reservada para a questão acima -	

ARQUIVOS	ARQUIVOS
Como podemos escrever em dados de modo binário?	Para escrever (salvar) dados em arquivos binários, usamos a função fwrite. O protótipo dessa função pode ser simplificado por: ***int fwrite (void* p, int tam, int nelem, FILE* fp);*** O primeiro parâmetro dessa função representa o endereço de memória cujo conteúdo se deseja salvar em arquivo. O parâmetro tam indica o tamanho, em bytes, de cada elemento, e o parâmetro nelem indica o número de elementos. Por fim, passa-se o ponteiro do arquivo binário para o qual o conteúdo de memória será copiado.
Como podemos ler em dados de modo binário?	A função para a leitura (recuperação) dos dados de arquivos binários é análoga, sendo que agora o conteúdo do disco é copiado para o endereço de memória passado como parâmetro. O protótipo da função pode ser dado por: ***int fread (void* p, int tam, int nelem, FILE* fp);***
Como podemos percorrer um arquivo de modo binário para recuperar somente as informações que desejamos?	Podemos alterar a posição do cursor do arquivo, o que permite posicioná-lo para ler uma determinada informação. A função que permite movimentar o cursor do arquivo tem o seguinte protótipo: ***int fseek (FILE* fp, long offset, int origem);*** O primeiro parâmetro indica o arquivo no qual estamos reposicionando o cursor. O segundo parâmetro indica quantos bytes iremos avançar, e o terceiro indica em relação a que posição estamos avançando o cursor: em relação à posição corrente (SEEK_CUR), em relação ao início do arquivo (SEEK_SET) ou em relação ao final do arquivo (SEEK_END). Para exemplificar, vamos considerar a existência de pontos no espaço 3D. Vamos, então, escrever uma função que, dado um ponteiro para esse arquivo aberto para leitura, faça a captura do i-ésimo ponto armazenado. Uma possível implementação dessa função é mostrada a seguir. ***Ponto le_ponto (FILE* fp, int i)*** ***{*** ***Ponto p;*** ***fseek (fp, i*sizeof (Ponto), SEEK_SET);*** ***fread (&p, sizeof (Ponto), 1, fp);*** ***return (p);*** ***}***
- reservada para a questão acima -	
- reservada para a questão acima -	