Parâmentro

Escreva uma função para um caixa de banco, que recebe um valor inteiro e determina o número de notas de 100, 50, 10, 5 e 1 reais necessário para pagar a quantia . Faça de forma que o número de notas usado seja o menor possível, retornando as quantidades de notas por referência.

Escreva uma função que receba um tempo em total de segundos desde a meia-noite, retornando por referência o tempo em horas, minutos e segundos correspondentes.

Escreva uma função que receba um vetor de inteiros, mais a quantidade de valores presentes nele, e retorne um número inteiro correspondente à quantidade de valores que aparecem mais de uma vez. Exemplo: se o vetor for [1, 2, 3, 4, 5, 3, 6, 7, 2, 8], o retorno será 2 (2 números se repetem, 2 e 3).

Strings,Structs e Unions

Escreva uma função chamada **minha_substring**, que recebe duas strings como parâmetro e verifica se a segunda é substring da primeira, retornando a **posição** onde ela começa na primeira. Caso não seja encontrada, retorne **-1:**

int minha_substring(char str1[], char str2[])

Escreva funções para complementar o código desenvolvido em aula de manipulação de frações através de struct:

- `Fracao soma(Fracao a, Fracao b)`
- `Fracao multiplica(Fracao a, Fracao b)`
- Você pode tentar simplificar a fração resultante, se quiser (dá um pouco mais de trabalho)

Escreva uma função chamada minha_substring, que recebe duas strings como parâmetro e verifica se a segunda é substring da primeira, retornando a posição onde ela começa na primeira. Caso não seja encontrada, retorne -1:

int minha_substring(char str1[], char str2[])

Arquivos

1. Implemente um programa em C que lê uma string e num número do teclado e escreve em um arquivo de texto.

2. Implemente um programa de teste com funções para ler e escrever registros (structs) em um arquivo binário em C. Conside a struct a seguinte:

```
typedef struct {
    char nome[64];
    int idade;
    float nota;
} aluno_t;
```

Implemenete as funções:

- int aluno_salva(const char *arquivo, aluno_t *aluno);
- int aluno_carrega(const char *arquivo, aluno_t alunos[], int len);

Descrição das funções:

- aluno_salva: Recebe o nome de um arquivo binário e uma referência de aluno_t. Deve abrir o arquivo especificado no modo de adição binária ("ab"), escrever a struct aluno_t no final do arquivo e fechar o arquivo. Retorna um inteiro indicando sucesso ou falha na operação.
- aluno_carrega: Recebe o nome de um arquivo binário, um array de aluno_t para armazenar os dados lidos, e o tamanho desse array (len). Deve abrir o arquivo especificado no modo de leitura binária ("rb") e ler as structs a partir do início carregando os dados no array fornecido, limitando-se ao tamanho len. Retorna o número de registros lidos com sucesso.

Manipulação de bits

Exercício 1: Usando operadores lógicos e de deslocamento, tente imaginar como implementar as seguintes operações.

Crie uma função para cada uma (a função deve retornar o valor alterado):

- Zera todos os bits: *unsigned int clear(unsigned int val)*
- Seta um único bit: *unsigned int setbit (unsigned int x, int bit)*
- Resseta um único bit: *unsigned int clearbit (unsigned int x, int bit)*
- Inverte o valor de um único bit: *unsigned int invertBit (unsigned int x, int bit)*
- Retorna o estado de um determinado bit: *int testBit (unsigned int x, int bit)* (retorna 0 ou 1)

Exercício 2: Crie um programa completo que defina uma varíavel *colour* como um *unsigned int*:

Em uma repetição:

- Mostre o valor atual na tela, em binário e hexa;
- Permita ao usuário escolher o componente de cor desejado (*red*, *green* ou *blue*);
- Pergunte o valor novo e armazene no campo de bits correto.