UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
CURSO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMAÇÃO DE COMPUTADORES

# LABORATÓRIO 26

FUNÇÕES COM STRINGS E REGISTROS

## EXERCÍCIOS DE REVISÃO

VOCÊ DEVE ACOMPANHAR PARA OBTER INFORMAÇÕES COMPLEMENTARES

1. O que a significam as expressões abaixo?

   a. `*"queijo"`
   b. `"presunto"[2]`

2. Escreva uma função com o protótipo abaixo. Ela deve substituir todas as ocorrências do caractere c1 por c2 na string str, retornando o número de substituições feitas.

```
int subst(char * str, char c1, char c2);
```

3. Considere a seguinte declaração de registro:

```
struct candidato
{
   char nome[30];
   int  notas[3];
};
```

   a. Escreva uma função que receba um **registro candidato** como argumento e mostre seu conteúdo.

   b. Escreva uma função que receba o **endereço de um candidato** e mostre o conteúdo do registro.

   c. Escreva uma função que receba um **vetor de candidatos** e mostre todo o conteúdo do vetor.

   d. Existe uma diferença na **assinatura das funções** usadas para resolver os itens “b” e “c”?

# EXERCÍCIOS DE FIXAÇÃO

VOCÊ DEVE FAZER OS EXERCÍCIOS PARA FIXAR O CONTEÚDO

1. Considerando o código abaixo:

```
#include <iostream>
#include <array>
#include <string>
using namespace std;

const array<string, 4> Mes = { "Marco", "Abril", "Junho", "Julho" };

void preencher(array<double, 4> * vet);
void mostrar(array<double, 4> vet);

int main()
{
	array<double, 4> gastos;
	preencher(&gastos);
	mostrar(gastos);
	return 0;
}

void preencher(array<double, 4> * vet)
{
	for (int i = 0; i < vet->size(); i++)
	{
		cout << "Entre com o gasto de " << Mes[i] << ": ";
		cin >> vet->at(i);   // (*vet)[i];
	}
}

void mostrar(array<double, 4> vet)
{
	double total = 0.0;
	cout << "\nGastos\n";
	for (int i = 0; i < vet.size(); i++)
	{
		cout << Mes[i] << ": R$" << vet[i] << endl;
		total += vet[i];
	}
	cout << "Total de gastos: R$" << total << endl;
}
```

Faça uma versão do programa que use vetores comuns de `const char *` para as strings representando os meses, e use um vetor comum de `double`'s para os gastos.

2. Refaça o programa da questão anterior usando vetores comuns de `const char *` para as strings representando os meses, e use um registro cujo único membro é um vetor estático de `double`'s para os gastos.

**Dica**: esta organização é similar à usada pela classe `array`.

3. Complete o programa esqueleto abaixo, fornecendo as funções necessárias.

```
#include <iostream>
using namespace std;

const int TAM_NOME = 30;

struct aluno
{
	char nome[TAM_NOME];
	int  nivel;
};

// solicita e armazena informações de alunos:
// - encerrando ao preencher o vetor ou quando o usuário
//   entrar com uma linha em branco para o nome do aluno
// - a função retorna o número de alunos lidos
int PegarInfo(aluno va[], int n);

// mostra o conteúdo de um registro aluno
void Mostrar1(aluno a);

// mostra o conteúdo do registro aluno apontado
void Mostrar2(const aluno * pa);

// mostra o conteúdo do vetor de alunos
void Mostrar3(const aluno va[], int n);

int main()
{
	cout << "Tamanho da classe: ";
	int tam;
	cin >> tam;

	 // remove \n para uso do cin.getline
	cin.ignore();

	aluno * ptr = new aluno[tam];
	int inscritos = PegarInfo(ptr, tam);
	for (int i = 0; i < inscritos; ++i)
	{
		Mostrar1(ptr[i]);
		Mostrar2(&ptr[i]);
	}
	Mostrar3(ptr, inscritos);

	delete[] ptr;
	cout << "Feito!\n";

	return 0;
}
```

# EXERCÍCIOS DE APRENDIZAGEM

VOCÊ DEVE ESCREVER PROGRAMAS PARA REALMENTE APRENDER

1. Escreva uma função que receba uma string por parâmetro e retorne um valor inteiro indicando quantas palavras existem na string. No programa principal leia uma string do usuário para testar a função. Use `const` sempre que pertinente.

```
Entre com uma frase:
Hoje em dia é difícil conhecer tudo.
Existem 7 palavras nesta frase!
```

2. Construa um programa que leia uma lista de nomes completos. O programa deve encerrar a leitura apenas quando for digitado o nome "fim". Passe cada nome completo para uma função que deve colocar as primeiras letras de cada parte do nome para maiúsculo e o restante para minúsculo, como mostrado no exemplo abaixo. A função deve retornar um booleano indicando se alguma alteração foi feita no nome original. O programa principal deve usar o valor de retorno para indicar se o nome foi ou não corrigido. Observe que o programa não põe para maiúsculo os nomes que tenham dois ou menos caracteres, como os conectores "da" e "de".

```
Entre com os nomes dos aprovados (fim para encerrar):
Pedro henrique augustiNO
Pedro Henrique Augustino  <-- Corrigido
joão baldo de albuquerque
João Baldo de Albuquerque <-- Corrigido
RIVALDO VIANA DA CRUZ
Rivaldo Viana da Cruz     <-- Corrigido
Eveline da Silva Bastos
Eveline da Silva Bastos   <-- OK
fim
```

3. Crie uma função chamada FirstName e outra chamada LastName. As funções devem receber uma string de origem e uma string de destino. A string de origem deve conter o nome completo de uma pessoa. As funções devem copiar para a string de destino, respectivamente, o primeiro e o último nome da pessoa. No programa principal leia vários nomes completos e exiba os resultados das funções, como no exemplo abaixo.

```
Entre com os nomes a cadastrar ('.' para encerrar):
Júnior Lima da Silva
Silva, Júnior
Benedito Nogueira Ramalho
Ramalho, Benedito
Ricardo Amaral Neto
Neto, Ricardo
Jamile Osório de Paula
Paula, Jamile
.
```

4. Um supermercado vende uma unidade de um produto ao seu preço normal, mas a partir de uma certa quantidade de produtos, ela vende por um preço menor, chamado de preço de atacado. Crie um **registro produto** para guardar o preço normal de um produto, seu preço de atacado e a quantidade a partir da qual o preço de atacado deve ser utilizado.

   Construa uma função que receba **um registro** do tipo produto e a quantidade pedida desse produto e retorne o total a pagar. A função deve analisar se deve aplicar o preço normal ou o preço de atacado de acordo com a quantidade solicitada.

   Utilize essa função dentro de outra função, cuja tarefa é receber um **vetor de produtos** e um **vetor de pedidos** e retornar o total do pedido. A função principal deve receber os produtos e os pedidos através de dois arquivos texto separados. Os arquivos possuem respectivamente as informações dos produtos e as quantidades pedidas, seguindo o formato abaixo. O primeiro número de cada arquivo é sempre a quantidade de linhas de informação.

   Produtos.txt

```
6
N: 3.50 A: 2.95 Q: 5
N: 9.80 A: 8.90 Q: 3
N: 5.00 A: 4.00 Q: 5
N: 7.85 A: 6.95 Q: 2
N: 1.95 A: 1.50 Q: 8
N: 4.90 A: 4.70 Q: 6
```

   Pedido.txt

```
6
3
4
10
5
5
8
```

   (N)ormal, (A)tacado, (Q)uantidade necessária para o preço de atacado

5. Construa um registro para armazenar um conjunto de caracteres. O registro deve guardar um ponteiro para um vetor dinâmico de caracteres e o tamanho do vetor, como mostrado abaixo:

```
struct CharSet
{
    char * str;     // ponteiro para vetor dinâmico
    int tam;        // tamanho do vetor
};
```

   Agora escreva funções para:

   a. Atribuir uma constante string para um CharSet já existente
   b. Exibir uma string armazenada em um CharSet
   c. Retornar o tamanho de um CharSet
   d. Concatenar uma constante string em um CharSet

   Construa uma solução em que o vetor interno do CharSet cresça automaticamente sempre que necessário. Todas as funções devem receber o **endereço de um CharSet** e não uma cópia.