"Se não existisse C hoje estaríamos programando em Obol, Pasal e BASI".

Introdução ao C - Parte 2

Paulo Ricardo Lisboa de Almeida

DSBD

UFPR
UNIVERSIDADE FEDERAL DO PARANÁ

Vetores

Em C, um vetor pode ser instanciado da seguinte forma

```
tipo nomeVetor[TAMANHO];
```

Para acessar um elemento do vetor, utilize `[ ]`

Exemplo

```
nomeVetor[5] = 20; // a posição 5 do vetor recebeu o valor 20
```

Vetores

Atenção

Em C, todo vetor de tamanho N **começa em 0, e termina em N-1**.

C não impede o acesso a posições inválidas do vetor. Logo, você pode corromper a memória se cometer algum erro de acesso aos índices.

Exemplo

```c
#include<stdio.h>

int main(){
      int vetor[10];
      for(int i=0; i < 10; i++){
            vetor[i] = i*10;
            printf("vetor[%d] possui %d\n", i, vetor[i]);
      }

      return 0;
}
```

Define

A diretiva #define serve para definir um valor via macro

Formato - note que não termina em ;

#define NOME_MACRO VALOR_MACRO

Os defines não devem estar dentro de nenhum bloco (ex.: não deve estar dentro do main)

Dica: faça seus defines logo após dos #includes

O compilador (na verdade o pré-processador) busca todos os lugares do código onde existe NOME_MACRO e substitui por VALOR_MACRO antes de compilar

Define

#define NOME_MACRO VALOR_MACRO

Defines são especialmente úteis para definirmos constantes, ou quando estamos lidando com vetores

Atenção

> Defina o nome da macro em MAIÚSCULO, se for um nome composto, separe por underscores _

Exemplo

```c
#include<stdio.h>

#define M 10

int main(){
      int vetor[M];
      for(int i=0; i < M; i++){
            vetor[i] = i*10;
            printf("vetor[%d] possui %d\n", i, vetor[i]);
      }

      return 0;
}
```

Funções

Em C, uma função é declarada da seguinte forma

```
tipoRetorno nomeFuncao(tipo parametro1, tipo parametro2, …){
    //corpo da função
}
```

`tipoRetorno` é o tipo do dado que será retornado pela função, e pode ser qualquer tipo válido

Uma função que não retorna valor algum pode ainda ser definida como ***void***

Funções

```
tipoRetorno nomeFuncao(tipo parametro1, tipo parametro2, …){
    //corpo da função
}
```

Uma função pode ter quantos parâmetros forem necessários (inclusive zero)

Todo parâmetro é **passado por cópia** (o valor original não é alterado)

Você vai aprender como passar parâmetros por ponteiro nas disciplinas focadas em programação C

Nesse caso, a "variável original" pode ser modificada

Quando passamos vetores (veremos adiante) de certa forma estamos passando um ponteiro, e nesse caso **alteramos o valor original - vetores não são passados por cópia**

Exemplo

```c
#include<stdio.h>


void funcaoSemRetorno(int valor){
	printf("O parametro eh %d\n", valor);
}

int retornaMaior(int val1, int val2){
	if(val1 > val2){
		return val1;
	}else{
		return val2;
	}
}

int main(){
	int retorno;

	funcaoSemRetorno(15);
	retorno = retornaMaior(10,20);
	printf("%d\n", retorno);

	printf("Pode chamar uma funcao dentro de outra %d\n", retornaMaior(25,1));

	return 0;
}
```

Escopo

Não use variáveis globais

Exceto se você tiver um bom motivo para isso

Variáveis locais existem somente dentro de seu escopo e dos escopos filhos desse escopo

Deixam de existir quando saem do escopo

{ ... }

Exemplo

```c
#include<stdio.h>

int funcao(int var){
	return var+1;
}
//Existe uma variável var no main, e uma em funcao
//são variáveis diferentes, cada uma dentro de seu escopo
int main(){
	int var = 10;
	int retorno;

	retorno = funcao(var);

	printf("%d %d\n", var, retorno);
	if(var == 11){
		int aux = var + 1;
		printf("%d\n", aux);
	}
	//aux existe apenas dentro do escopo do if
	//tentar acessar aqui resulta em um erro
	//printf("%d\n", aux);
	return 0;
}
```

Vetores como parâmetro

Para passar um vetor como parâmetro de uma função, adicione `[ ]` no final do parâmetro para indicar que ele é um vetor

```
tipoRetorno nomeFuncao(tipoVetor nome[],…){
     //corpo da função
}
```

Reforçando: vetores não são passados por cópia!

Exemplo

```c
#include<stdio.h>

#define M 5

void imprimirVetor(int vet[], int tam){
	for(int i=0; i < tam; i++){
		printf("%d ", vet[i]);
	}
	printf("\n");
}
void modificarVetor(int vet[], int tam){
	for(int i=0; i < tam; i++){
		vet[i]++;
	}
}
int main(){
	int vetor[M];

	for(int i=0;i<M;i++){
		vetor[i] = i*2;
	}
	imprimirVetor(vetor, M);
	modificarVetor(vetor, M);
	imprimirVetor(vetor, M);

	return 0;
}
```

Exercícios

Faça os seguintes programas em C

1. Considere que o usuário vai digitar 3 inteiros entre 0 e 999. Crie uma função que receba esses 3 inteiros, e os retorna em um único inteiro. Por exemplo, se os inteiros são 500, 12 e 47, a função deve retornar 500012047. Chame essa função de compressor.
2. Faça uma função que faz o contrário do exercício 1. Recebe um inteiro "comprimido", e um vetor de três posições. A função deve armazenar os valores individuais nas posições do vetor. Chame a função de descompressor.
3. Crie uma função que recebe dois vetores de tamanho N, e um vetor de tamanho 2N. A função deve intercalar os dois vetores e armazenar o resultado no vetor de tamanho 2N.

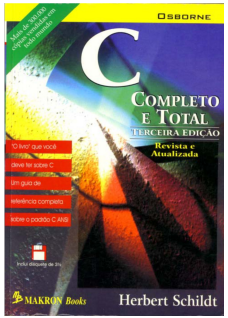
   Vetor1: 10 1 45 7
   Vetor2: 12 98 90 11
   VetorIntercalado: 10 12 1 98 45 90 7 11

4. Crie uma função que recebe dois vetores float, sendo que esses vetores são de mesmo tamanho e representam as coordenadas x e y, respectivamente, de um conjunto de N pontos. A função também deve receber a coordenada de um ponto P (coordenada x e coordenada y). Como resposta, a função deve retornar o índice do ponto dos vetores que está mais próximo de P.

Referências

www.open-std.org/jtc1/sc22/WG14/www/docs/n1256.pdf

H. Schildt. C completo e total. 1996



Licença