Linguagem de Programação I - ESP201

Profª Ana Carolina Sokolonski

Bacharelado em Sistemas de Informação
Instituto Federal de Ciência e Tecnologia da Bahia
Campus de Feira de Santana

carolsoko@ifba.edu.br

April 8, 2025

Ponteiros

Ponteiros

Ponteiros

- Ponteiros é um dos recursos mais poderosos da linguagem C.
- Ponteiros ou apontadores são variáveis que armazenam o **endereço de memória** de outras variáveis. Dizemos que um ponteiro **"aponta"** para uma variável quando contém o endereço da mesma.
- Os ponteiros podem apontar para qualquer tipo de variável. Portanto, ponteiros podem apontar para int, float, double, char, etc.

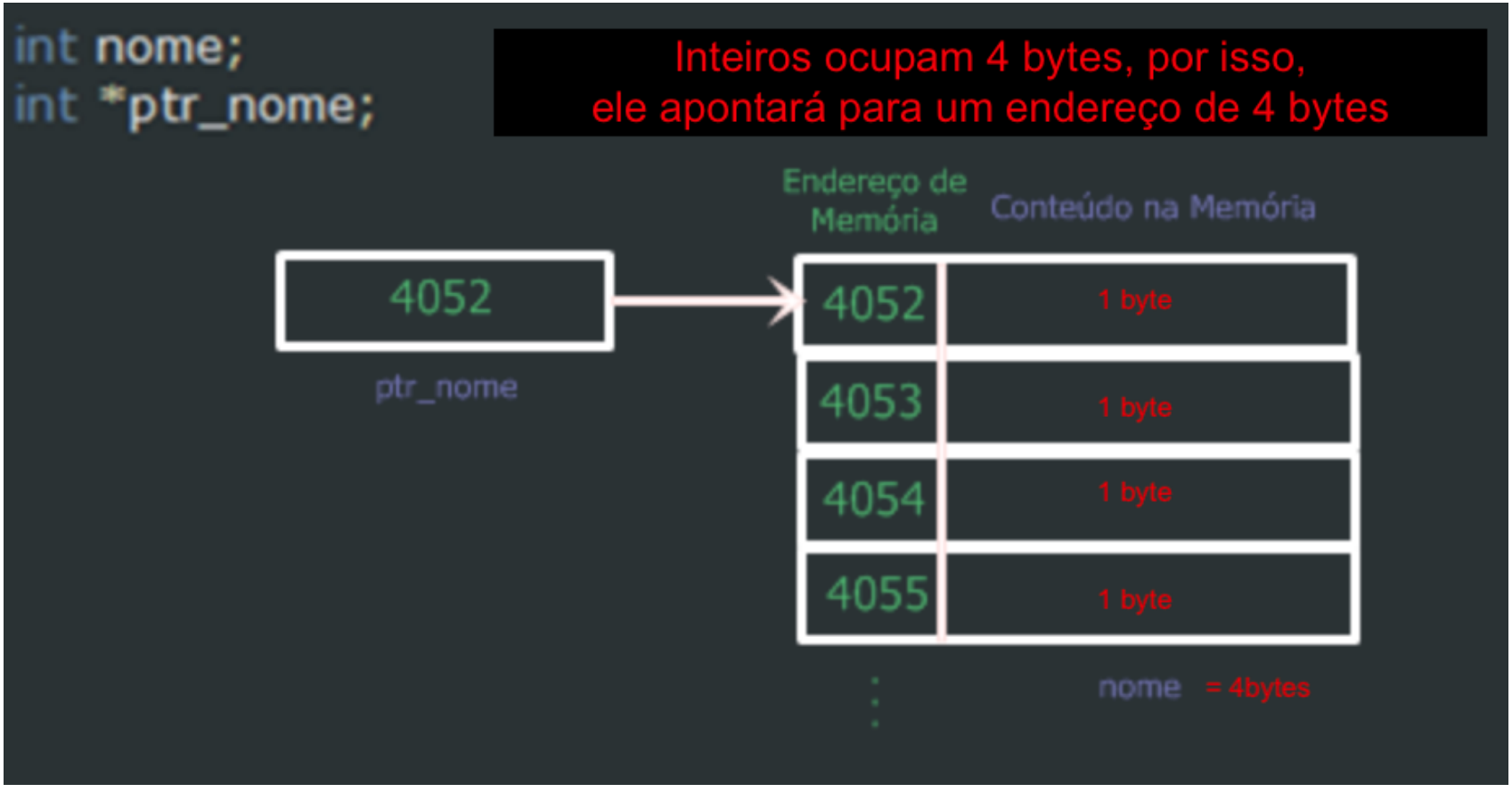
Declaração de um ponteiro - Sintaxe

Para declarar um ponteiro em C basta colocarmos um asterisco - * - antes do nome da variável:

```
int *nome;
struct alunoDaTurma *aluno1;
float *idade;
```

Declaração de um ponteiro - Sintaxe

Ponteiros armazenam os endereços das variáveis:



Vantagens dos Ponteiros

Vantagens dos Ponteiros

1. Ponteiros são muito úteis quando uma variável tem que ser acessada em diferentes partes de um programa. Neste caso, o código pode ter vários ponteiros espalhados por diversas partes do programa, **"apontando"** para a variável que contém o dado desejado. Sem precisar percorrer uma estrutura de dados completa, como o caso de um vetor ou matriz [Schildt e Mayer 1997].

Vantagens dos Ponteiros

Vantagens dos Ponteiros

2. Caso o dado apontado seja alterado, não há problema algum. Todas as partes do programa que têm um ponteiro que aponta para **o endereço** onde reside o dado estarão com acesso ao dado atualizado. O dado é modificado dentro do endereço, o seu endereço permanece o mesmo.

Vantagens dos Ponteiros

Vantagens dos Ponteiros

3. Com o uso de ponteiros podemos crescer nossas variáveis de acordo com a necessidade, não precisamos fazer como no vetor, que alocamos um vetor de 100 posições para, eventualmente ocupar apenas 10 posições, porque no futuro podemos precisar das outras 90 posições. Podemos alocar apenas o que precisamos à medida que vamos precisando.

Desvantagens dos Ponteiros

Desvantagens dos Ponteiros

1. O uso do ponteiro requer **domínio de programação (BONS PROGRAMADORES)**, pois qualquer erro pode fazer você perder o endereço apontado, e sem referência àquele endereço, a informação é perdida na memória.

Inicializando um ponteiro

- Para atribuir um valor a um ponteiro recém-criado poderíamos igualá-lo a um endereço de memória.
- Como podemos descobrir os endereços de memória de cada variável?
- Seria muito difícil saber o endereço de cada variável que usamos, mesmo porque estes endereços são determinados pelo compilador na hora da compilação e realocados a cada execução.
- o ideal é deixar que o próprio compilador informe este endereço. Para tanto, usamos o &

Declaração de um ponteiro - Sintaxe

Inicializando ponteiros, usando o **&**

```
int contador=20;
int *ponteiro;
ponteiro = &contador;
```

Declaração de um ponteiro - Sintaxe

Uma vez que fizemos $ponteiro_idade = \&idade$ a expressão $*ponteiro_idade$ é equivalente à própria variável idade (conteúdo da variável). Isto significa que, se quisermos mudar o valor de idade para 70, basta fazer $*ponteiro_idade = 70$.

```
int idade, *ponteiro_idade;
idade = 40; //inicializando a idade com o valor 40
ponteiro_idade = &idade; // apontando ponteiro_idade para idade
*ponteiro_idade = 70; //modificando o valor da variável idade para 70
```

Passagem de Structs por Referência

Passagem de Structs por Referência

Para passar uma struct por referência, deve-se passar um ponteiro para a struct, como no exemplo a seguir.

```
void SetPessoa(Pessoa *P, int idade, float peso, float altura){
  (*P).Idade = idade; // o campo pode ser acessado desta forma
   P->Peso = peso; // ou desta
   P->Altura = altura;
}
```

Passagem de Structs por Referência

```c
#include <stdio.h>
typedef struct pessoa{
    float Peso;
    int Idade;
    float Altura;
} Pessoa;

void ImprimePessoa(Pessoa P){
  printf("Idade: %d  Peso: %5.2f Altura: %5.2f\n", P.Idade, P.Peso, P.Altura);
}

void SetPessoa(Pessoa *P, int idade, float peso, float altura){
    (*P).Idade = idade;  // o campo pode ser acessado desta forma
    P->Peso = peso;      // ou desta
    P->Altura = altura;
}

int main(){
    Pessoa Joao;
    SetPessoa(&Joao, 15, 70.5, 1.75);
    ImprimePessoa(Joao);
    return 0;
}
```

Referências

Referências


SCHILDT, H.; MAYER, R. *C completo e total*. [S.l.]: Pearson University, 1997. ISBN 9788534605953.