Programação Estruturada
Carolina Aguilar

Passagem de Parâmetro por Referência

Aula 16 – 2020.1

Passagem de Parâmetro por Referência

- Operadores utilizados
  - Operador unário & (“endereço de)
  - Operador unário * (“conteúdo de)

Passagem de Parâmetro por Referência

- Operador unário *
  - Na declaração de uma variável indica que ela irá armazenar um endereço de memória. Deste modo, em vez da variável armazenar um dado de um determinado tipo, ela irá armazenar um ponteiro para o dado. Isto significa que a variável irá guardar o endereço do local onde o dado de um determinado tipo está guardado.
  - Em uma expressão, quando aparece junto com uma variável que foi declarada com sendo um ponteiro (declarada com tipo *), significa o conteúdo do endereço.

Passagem de Parâmetro por Referência

- Operador unário &
  - Utilizamos sempre ele na função scanf( ). Quando usamos **&nomeVariavel** indicamos que estamos passando para a função o endereço de memória de uma variável. Deste modo estamos passando "onde" (endereço de memória) ela irá armazenar o dado.

Passagem de Parâmetro por Referência

- Declaração:
  - Forma geral

    tipo *nomeVariavel;
  - Exemplo

    int *end;
- Inicialização:
  - Forma geral

    nomeVariavel = &outraVariaveltipo;
  - Exemplo

    int idade;

    int *end;

    end = &idade;

Passagem de Parâmetro por Referência

- Declaração e inicialização:
  - Forma geral

    tipo *nomeVariavel = &outraVariaveltipo;
  - Exemplo

    int idade;

    int *end = &idade;

Passagem de Parâmetro por Referência

- Exemplo operadores unários * e &:

```
int main()
{
   int idade;
   int *end;
   idade = 18;
   end = 21;          //ERRADO!! end não pode armazenar um número inteiro
   end = &idade;   /*CORRETO end armazena o endereço de memória de um dado do
                      tipo int. */
   *end = 21;       /*CORRETO. Como end aponta para a área de memória da variável
                      idade. *end significa conteúdo do endereço. Logo o valor 21 será
                      armazenado na variável idade */
   return 0;
}
```

- Observações:
  - **&idade** significa endereço de memória da variável idade.

Passagem de Parâmetro por Referência

- Atenção:

```
int main()
{
   int idade;
   int *end;
   idade = 18;
   *end = 21;     /* ERRADO. Como end não foi inicializada com nenhum endereço de
                     memória, ou seja, está com NULL (endereço nulo), não podemos
                     guardar o valor 21 em um endereço inexistente */
   return 0;
}
```

Passagem de Parâmetro por Referência

- Exemplo:

  Faça um programa, utilizando a função abaixo, que leia 2 números inteiros, calcule e exiba a soma e o produto dos números.

  Faça a função **calculos** que recebe como parâmetros 2 números inteiros, calcule a soma e o produto dos números. Esta função deverá retornar a soma dos números e irá guardar o produto dos números na variável cujo endereço é fornecido na chamada da função.

Passagem de Parâmetro por Referência

- Exemplo – Solução ERRADA. Solução sem utilizar passagem de parâmetro por referência:

```
#include<stdio.h>
int calculos (int n1, int n2)
{
    int soma, prod;
    soma = n1 + n2;
    prod = n1 * n2;
    return soma;
}
int main()
{
    int num1, num2, somanum, prodnum;
    printf("Digite 2 numeros inteiros: ");
    scanf("%d%d", &num1, &num2);
    somanum = calculos(num1, num2);
    printf("\nSoma: %d\tProduto: %d", somanum,prodnum);
    return 0;
}
```

Passagem de Parâmetro por Referência

- Exemplo – Solução ERRADA. Solução sem utilizar passagem de parâmetro por referência:
  - A função **calculos**, calcula corretamente a soma e o produto dos números.
  - Como pelo **return** só podemos retornar um único dado do tipo especificado, a função retornou corretamente a soma dos números
  - O produto dos números foi perdido quando a função **calculos** foi encerrada.
  - Ao retornar para a função main( ), o valor retornado foi armazenado na variável **somanum**.
  - Entretanto nenhum valor foi armazenado na variável **prodnum**. Ela está com lixo.
  - Gostaríamos que ao término da execução da função cálculos, o produto dos números estivesse na variável **produm**.

Passagem de Parâmetro por Referência

- Exemplo – Solução CORRETA. Solução utilizando passagem de parâmetro por referência:

```
#include<stdio.h>
int calculos (int n1, int n2, int *prod)
{
    int soma;
    soma = n1 + n2;
    *prod = n1 * n2;
    return soma;
}
int main()
{
    int num1, num2, somanum, prodnum;
    printf("Digite 2 numeros inteiros: ");
    scanf("%d%d", &num1, &num2);
    somanum = calculos(num1, num2, &prodnum);
    printf("\nSoma: %d\tProduto: %d", somanum,prodnum);
    return 0;
}
```

Passagem de Parâmetro por Referência

- Exemplo – Solução CORRETA. Solução utilizando passagem de parâmetro por referência:
  - A função **calculos**, irá receber o endereço de memória da variável **prodnum**, guardando-o na variável **prod**.
  - A variável **prod** irá apontar para o local onde o dado (produto dos números) deverá ser armazenado.
  - A função **calculos**, irá guardar o resultado do produto dos números no endereço de memória da variável **prodnum** .
  - Deste modo, a variável **calculos**, variável da função main( ), é que irá armazenar o resultado do produto dos números.
  - O dado não será perdido ao término da execução da função **calculos** pois ele está armazenado na variável **prodnum**.

Exercícios

1. Faça apenas as duas funções auxiliares abaixo.
   a) Faça a função le_valida_tipo que leia e valide o tipo do produto (1-fruta, 2-legume, 3-verdura). A função deverá retornar o tipo validado.
   b) Faça uma função, utilizando a função acima, que receba a quantidade de produtos comercializados por uma loja como parâmetro. A função deverá ler para cada produto o código e o tipo (1-fruta, 2-legume, 3-verdura), descobrir a quantidade de produtos que são do tipo 1, a quantidade de produtos que são do tipo 2 e a quantidade de produtos que são do tipo 3. Para isto a função irá guardar as quantidades nas variáveis cujos endereços são fornecidos na chamada da função.

Exercícios

2. Faça um programa, utilizando a função abaixo, para processar os 40 alunos de uma turma. Para cada aluno o programa deverá ler a matrícula e as duas notas, calcular a média e exibir a matrícula, a média e a situação do aluno ("aprovado": média >= 5,0 ou "reprovado": média < 5,0). Ao final o programa deverá exibir a quantidade de alunos aprovados e a quantidade de alunos reprovados da turma. Faça uma função que receba a média de um aluno, exiba a sua situação e contabilize este aluno nos aprovados ou nos reprovados. A função irá contabilizar os aprovados e os reprovados nas variáveis cujos endereços são fornecidos na chamada da função.

Exercícios

3. Faça um programa, utilizando a função abaixo, que exiba o maior salário de cada departamento de uma empresa e quantos funcionários ganham o maior salário do departamento. Para cada departamento, o programa deverá ler o código do departamento e a quantidade de funcionários, e para cada funcionário, a matrícula e o salário. Término da leitura dos departamentos: código do departamento = 0.
   a) Faça a função **um_departamento** para processar os funcionários de um departamento. Esta função deverá receber como parâmetro a quantidade de funcionários do departamento, ler os dados de cada funcionário, descobrir o maior salário do departamento e quantos funcionários ganham este maior salário, armazenando-os nas variáveis cujos endereços são fornecidos na chamada da função.

Fim