Estácio

ARA0098 - ESTRUTURA DE DADOS EM C

PROFESSOR: LUCAS SAMPAIO LEITE

E os nossos exercícios?

1. Escreva um programa que declare um inteiro, um double e um char, e ponteiros para inteiro, double, e char. Associe as variáveis aos ponteiros (use &). Modifique os valores de cada variável usando os ponteiros. Imprima os valores das variáveis antes e após a modificação.
2. Escreva um programa que contenha duas variáveis inteiras. Compare seus endereços e exiba o maior endereço.
3. Escreva um programa que contenha duas variáveis inteiras. Leia essas variáveis do teclado. Em seguida, compare seus endereços e exiba o conteudo do maior endereço.
4. Faça as 4 primeiras questões deste game: https://www.codingame.com/playgrounds/24988/programacao-c/praticando-ponteiros-e-funcoes

Revisão funções

- **Definição:** Conjunto de comandos agrupados em um **bloco**, que recebe um **nome** e através deste pode ser **evocado**.

Revisão funções

- **Sintaxe:**

```
tipo_da_funcao  NomeDaFuncao (Lista_de_Parametros)
{
// corpo da função
}
```

A **Lista_de_Parametros,** também é chamada de Lista_de_Argumentos, é opcional.
tipo_da_função é o tipo do retorno da função.

Revisão funções

- **Porque usar funções?**
  - Para permitir o reaproveitamento de código já construído(por você ou por outros programadores);
  - Para evitar que um trecho de código que seja repetido várias vezes dentro de um mesmo programa;
  - Para permitir a alteração de um trecho de código de uma forma mais rápida. Com o uso de uma função é preciso alterar apenas dentro da função que se deseja;
  - Para que os blocos do programa não fiquem grandes demais e, por conseqüência, mais difíceis de entender;
  - Para facilitar a leitura do programa-fonte de uma forma mais fácil;
  - Para separar o programa em partes(blocos) que possam ser logicamente compreendidos de forma isolada.

Revisão funções

- **Exemplo:**
  - Em primeiro lugar, imaginemos que você necessite várias vezes em seu programa imprimir a mensagem "Pressione a tecla ENTER" e esperar que o usuário tecle ENTER, caso o usuário tecle algo diferente o programa deve imitir um BEEP.
  - Você pode fazer um laço de WHILE sempre que isto fosse necessário.
  - Uma alternativa é criar uma função. Com o uso de funções, este processo de repetição fica simplificado.

Revisão funções

funcoes.c > ...

```c
#include <stdio.h>

void EsperaEnter(){  // Definição da função "EsperaEnter"
    int tecla;
    printf("Pressione ENTER\n");
    do{
        tecla = getchar();
        if (tecla !=13){ // Se nao for ENTER
            printf("Digite ENTER\n");
        }
    } while(tecla != 13); // 13 e' o codigo ASCII do ENTER
}

void main(){
    EsperaEnter();      // Chamada da função definida antes
    // ..........
    EsperaEnter();      // Chamada da função definida antes
    // ..........
    EsperaEnter();      // Chamada da função definida antes
}
```

Revisão funções

- **Parâmetros de uma função:**
  - A fim de tornar mais amplo o uso de uma função, a linguagem C permite o uso de parâmetros. Este parâmetros possibilitam que se definida sobre quais dados a função deve operar.
  - Para definir os parâmetros de uma função o programador deve explicitá-los como se estive declarando uma variável, entre os parênteses do cabeçalho da função. Caso precise declarar mais de um parâmetro, basta separá-los por vírgulas.

Revisão funções

```c
void soma(float a, int b){  // basta separar por vírgulas
    float result;           // a declaração de variáveis é igual ao que
                            // se faz na função main
    result = a+b;
    printf("A soma de %.2f com %d é %.2f\n", a,b,result);
}
```

A função SOMA que possui dois parâmetros, sendo o primeiro um float e o segundo um int.

Revisão funções

funcoes.c > ...

```c
#include <stdio.h>

void soma(float a, int b){  // basta separar por vírgulas
    float result;     // a declaração de variáveis é igual ao que
                      // se faz na função main
    result = a+b;
    printf("A soma de %.2f com %d é %.2f\n", a,b,result);
}

int main(){
    int a;
    float b;

    a = 10;
    b = 12.3;
    soma(b,a);  // Chamada da função SOMA(12.3,10);

    return 0;
}
```

Os parâmetros são passados para uma função de acordo com a sua posição.

A função soma é chamada recebendo como parâmetros as variáveis "b" e "a", nesta ordem.

Revisão funções

```c
float soma(float a, int b){
    float result;
    result = a+b;
    return result;
}

int main(){
    int a;
    float b, result;

    a = 10;
    b = 12.3;
    result = soma(b,a);
    printf("A soma de %.2f com %d é %.2f\n", b,a,result);
    return 0;
}
```

A função soma agora retorna um float.

Revisão funções

- **Escopo de variáveis:**
  - Por escopo de uma variável entende-se o bloco de código onde esta variável é válida. Com base nisto, temos as seguintes afirmações:
    - As variáveis valem no bloco que são definidas;
    - As variáveis definidas dentro de uma função recebem o nome de variáveis locais;
    - Os parâmetros de uma função valem também somente dentro da função;
    - Uma variável definida dentro de uma função não é acessível em outras funções, MESMO ESTAS VARIÁVEIS TENHAM NOME IDÊNTICOS.

Revisão funções

```c
#include <stdio.h>
void func1(){
    int b;
    b = -100;
    printf("Valor de b dentro da função func1: %d\n", b);
}
void func2(){
   int b;
   b = -200;
   printf("Valor de b dentro da função func2: %d\n", b);
}
```

O que será impresso???

```c
int main(){
    int b;

    b = 10;
    printf("Valor de b: %d\n", b);
    b = 20;
    func1();
    printf("Valor de b: %d\n", b);
    b = 30;
    func2();
    printf("Valor de b: %d\n", b);
    return 0;
}
```

Revisão funções

```c
#include <stdio.h>
void func1(){
    int b;
    b = -100;
    printf("Valor de b dentro da função func1: %d\n", b);
}
void func2(){
   int b;
   b = -200;
   printf("Valor de b dentro da função func2: %d\n", b);
}
```

```c
int main(){
    int b;

    b = 10;
    printf("Valor de b: %d\n", b);
    b = 20;
    func1();
    printf("Valor de b: %d\n", b);
    b = 30;
    func2();
    printf("Valor de b: %d\n", b);
    return 0;
}
```

```
Valor de b: 10
Valor de b dentro da função func1: -100
Valor de b: 20
Valor de b dentro da função func2: -200
Valor de b: 30
```

Revisão funções

- **Passagem de parâmetro por valor:**
  - Nesta modalidade, a chamada da função passa o valor do parâmetro para a função. Desta forma, alterações do parâmetro dentro da função não afetarão a variável usada na chamada da função.

```c
#include <stdio.h>

void zera(float a){
    a = 0;
}

void main(){
    float f;
    f = 20.7;
    zera(f);
    printf("%d", f);

}
```

Revisão funções

- **Passagem de parâmetro por valor:**
  - Nesta modalidade, a chamada da função passa o valor do parâmetro para a função. Desta forma, alterações do parâmetro dentro da função não afetarão a variável usada na chamada da função.

O valor impresso será 20.7 pois o parâmetro da função foi passado por valor.

```c
#include <stdio.h>

void zera(float a){
    a = 0;
}

void main(){
    float f;
    f = 20.7;
    zera(f);
    printf("%.1f", f);
}
```

Revisão funções

- **Passagem de parâmetro por referência:**
  - Para permitir a alteração da variável usada como parâmetro é preciso passar o endereço da variável, caracterizando desta forma uma passagem por referência.
  - Na chamada da função deve-se usar o operador & antes do nome da variável;
  - No cabeçalho da função, declarar o parâmetro como um ponteiro;
  - Dentro da função, deve-se usar o operado de indireção * para alterar o conteúdo da variável.

```c
#include <stdio.h>

void zera(float *a){
    *a = 0;
}

void main(){
    float f;
    f = 20.7;
    zera(&f);
    printf("%.1f", f);

}
```

Revisão funções

- **Passagem de vetores por parâmetros.**
  - A passagem de vetores por parâmetro é sempre por referência.
  - No cabeçalho da função que recebe um vetor, há duas formas de definir o parâmetro que é um vetor.

```
void zeraVet(float V[10], int qtd){
    int i;
    for(i=0;i<qtd;i++)
        V[i] = 0.0;
}

void main(){
    int i;
    float vet[10];

    zeraVet(vet,10);
    for(i=0;i<10;i++)
       printf("%f ", vet[i]);
}
```

Revisão funções

- **Passagem de vetores por parâmetros.**
  - A passagem de vetores por parâmetro é sempre por referência.
  - No cabeçalho da função que recebe um vetor, há duas formas de definir o parâmetro que é um vetor.

```c
void zeraVet(float v[10], int qtd){
    int i;
    for(i=0;i<qtd;i++)
        v[i] = 0.0;
}

void main(){
    int i;
    float vet[10];

    zeraVet(vet,10);
    for(i=0;i<10;i++)
       printf("%f ", vet[i]);
}
```

Revisão funções

- **Passagem de vetores por parâmetros.**
  - A passagem de vetores por parâmetro é sempre por referência.
  - No cabeçalho da função que recebe um vetor, há duas formas de definir o parâmetro que é um vetor.

```
void zeraVet(float *v, int qtd){
    int i;
    for(i=0;i<qtd;i++)
        v[i] = 0.0;
}

void main(){
    int i;
    float vet[10];

    zeraVet(vet,10);
    for(i=0;i<10;i++)
       printf("%f ", vet[i]);
}
```

Exercícios

1. Elabore uma função que receba três notas de um aluno como parâmetro e uma letra. Se a letra for ‘A’, a função deve calcular a média aritmética das notas do aluno; se a letra for ‘P’, deverá calcular a média ponderada, com pesos 5, 3 e 2. Retorne a média calculada para o programa principal.
2. Escreva uma função que, dado um número real passado como parâmetro, retorne a parte inteira e a parte fracionária desse número por referência.

Exercícios

```c
#include <stdio.h>
#include <stdlib.h>

float media(float n1, float n2, float n3, char op) {
    if (op == 'A') {
        return (n1 + n2 + n3) / 3;
    } else if (op == 'P') {
        return (n1*5 + n2*3 + n3*2) / 10;
    }
}
```

```c
int main(){
    float nota1, nota2, nota3, calc;
    char op;

    printf("Digite a opção [A ou P]: ");
    scanf("%c", &op);
    printf("Digite as notas: \n");
    scanf("%f %f %f", &nota1, &nota2, &nota3);

    calc = media(nota1, nota2, nota3, op);

    printf("Media= %f", calc);

    return 0;
}
```

Exercícios

```c
#include <stdio.h>

void separaNumero(float num, int *x, float *y) {
    *x = (int)num;
    *y = num - *x;
}

int main(){
    float num, frac;
    int inteira;

    scanf("%f", &num);
    separaNumero(num, &inteira, &frac);
    printf("Número %f: %i e %f \n", num, inteira, frac);

    return 0;
}
```

Exercícios para casa...

1. Faça um programa, com uma função que necessite de três argumentos, e que forneça a soma desses três argumentos. Utilize passagem por referência.
2. Faça um programa, com uma função que necessite de um argumento. A função retorna o valor de caractere ‘P’, se seu argumento for positivo, e ‘N’, se seu argumento for zero ou negativo.
3. Faça um programa para imprimir de acordo com a imagem abaixo para um n informado pelo usuário. Use uma função que receba um valor n inteiro e imprima até a n-ésima linha.

Desafio...

- Leia um número decimal (até 3 dígitos) e escreva o seu equivalente em numeração romana. Utilize funções para obter cada dígito do número decimal e para a transformação de numeração decimal para romana (1 = I, 5 = V, 10 = X, 50 = L, 100 = C, 500 = D, 1.000 = M; e utilize um vetor guardando a tradução para cada um dos dígitos).

Dúvidas???