Comandos em C (cont.)

Operador ternário: ?

- O operador condicional possui uma opção um pouco estranha. É o único operador C que opera sobre três expressões. Sua sintaxe geral possui a seguinte construção:

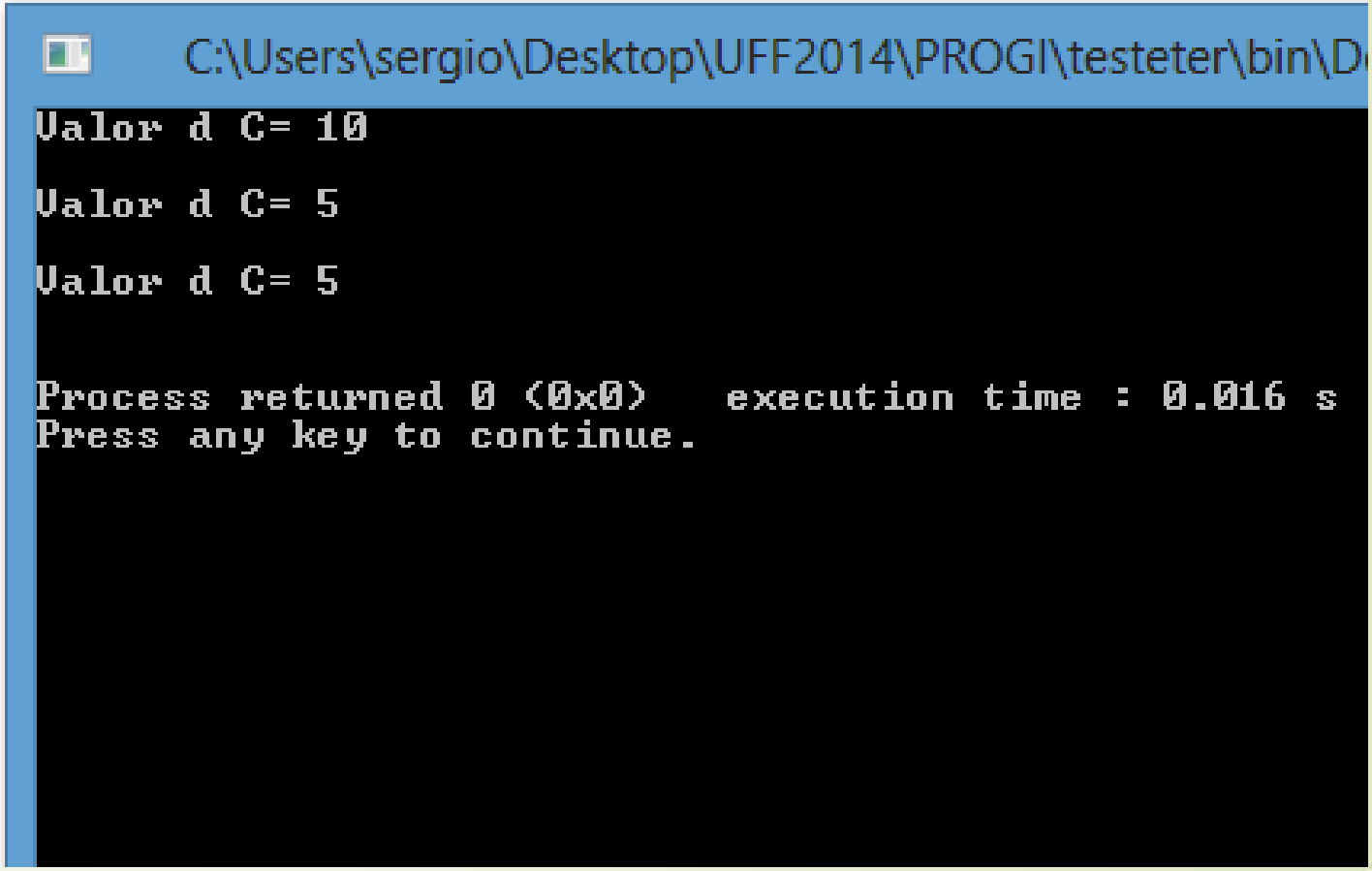
- Exp1 ? Exp2 : Exp3

- Exp1 é avaliada primeiro. Se o seu valor for diferente de zero (logo verdadeira), a exp2 será avaliada e o seu resultado será o valor da expressão condicional como um todo. Se exp1 for zero, a exp3 será avaliada e será o valor da expressão condicional como um todo.

- Na expressão **max**= (a>b) **? a : b**; a variável que contém o maior valor numérico entre a e b será atribuída a **max**.

Operador ternário: ?

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int a, b, c;
5  int main()
6  {
7      a=10;
8      b=5;
9      c= (a>b)? a: b;
10     printf("Valor d C= %d\n\n", c);
11     c= (a<b)? a: b;
12     printf("Valor d C= %d\n\n", c);
13     c= (a=b)? a: b;
14     printf("Valor d C= %d\n\n", c);
15     return 0;
16 }
17
```



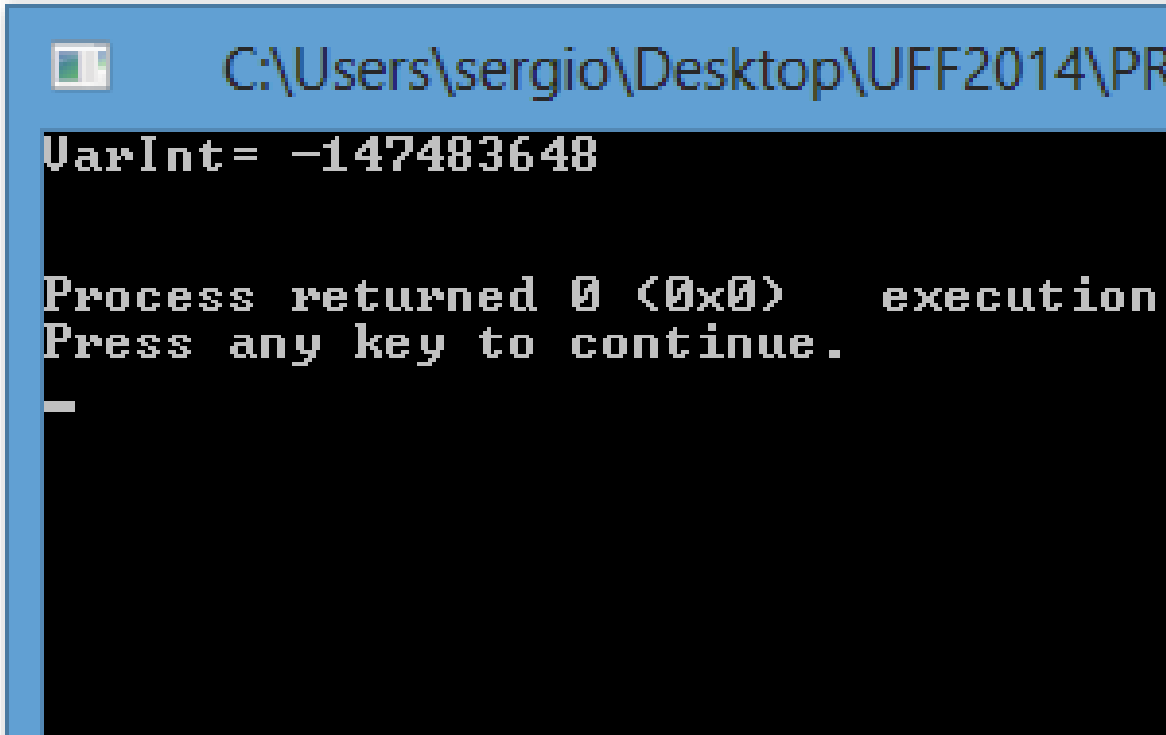
Conversão de tipos

- **Conversão automática de tipo**: quando 2 ou mais operandos de diferentes tipos se encontram em uma mesma expressão, o conteúdo da variável de menor tamanho é convertido ao tipo da variável de maior tamanho.
- Algumas vezes esta conversão não faz exatamente o que pensamos.

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   int VarInt=2000000000;
5   int Dez=10;
6
7   int main()
8   {
9    VarInt= (VarInt*Dez)/Dez;
10   printf("VarInt= %d\n\n", VarInt);
11   return 0;
12  }
13
```

Conversão automática de tipo

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int VarInt=2000000000;
5  int Dez=10;
6
7  int main()
8  {
9   VarInt= (VarInt*Dez)/Dez;
10  printf("VarInt= %d\n\n", VarInt);
11  return 0;
12 }
13
```



Conversão explícita

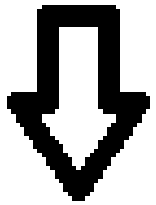
- O Operador de molde consiste em escrever o nome do tipo desejado entre parênteses e , em seguida, o valor ou a expressão a ser avaliada. O resultado é a expressão convertida para o tipo especificado.

- Sintaxe:

- (tipo desejado)variável ou (tipo desejado)expressão

Conversão explícita

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   int VarInt=2000000000;
5   int Dez=10;
6
7   int main()
8   {
9    VarInt= ( (double)VarInt*Dez)/Dez;
10   printf("VarInt= %d\n\n", VarInt);
11   return 0;
12  }
13
```



```
C:\Users\sergio\Desktop\
VarInt= 2000000000

Process returned 0 (0x0)
Press any key to continue.
```

Laço **do-while**

- É um laço bastante similar ao **while**. Ele é utilizado em situações em que é necessário executar o corpo do laço um primeira vez e, depois, avaliar a expressão de teste e criar um ciclo repetido.
- Sintaxe:

```
do
{
   Instrução
   Instrução
} while (teste)
```

Laço **do-while**

```
1  #include<stdio.h>
2  int main()
3  {
4  int n;
5  int soma = 0;
6  do
7  {
8     printf("Digite um numero positivo para ser somado ou negativo para sair: ");
9     scanf("%d", &n);
10    if( n >= 0 )
11       soma = soma + n;
12 }while( n >= 0 );
13
14 printf("A soma eh %d\n", soma);
15 return 0;
16 }
17
```

```
C:\Users\sergio\Desktop\UFF2014\PROG\testeter\bin\Debug\testeter.exe
Digite um numero positivo para ser somado ou negativo para sair: 1
Digite um numero positivo para ser somado ou negativo para sair: 2
Digite um numero positivo para ser somado ou negativo para sair: 3
Digite um numero positivo para ser somado ou negativo para sair: 4
Digite um numero positivo para ser somado ou negativo para sair: 0
Digite um numero positivo para ser somado ou negativo para sair: -9
A soma eh 10

Process returned 0 (0x0)   execution time : 17.389 s
Press any key to continue.
```

Comandos **break** e **continue**

- Os comandos **break** e continue são instruções que devem pertencer ao corpo de um laço **for**, **while** ou **do-while** e não podem ser utilizados em outras partes de um programa.
- O comando **break** também é utilizado no comando **switch**.

- O comando **break** causa uma saída imediata de um laço, após isto o comando passa para a próxima instrução.
- O comando **continue** força a próxima iteração do laço e pula o código que estiver abaixo. Nos laços **while** e **do-while**, um continue faz com que o controle do programa avalie imediatamente a expressão de teste do laço.
- No **for** o continue força o incremento da expressão e depois a sua avaliação.

break dentro do for

```c
#include <stdio.h>

int main()
{
        int count,
            numero;

        for(count=1 ; count<=1000000 ; count++)
            if((count%11==0) && (count%13==0) && (count%17==0))
            {
                numero=count;
                break;
            }


        printf("O numero e: %d", numero);

}
```

continue dentro do for

```c
#include <stdio.h>

int main()
{
        int count,
            soma;

        for(count=1, soma=0 ; count<=100 ; count++)
        {
            if( count%5 ==0)
                continue;

            soma = soma + count;
        }


        printf("Soma %d", soma);

}
```

continue dentro do do-while

```c
#include <stdio.h>

int main ()
{
   /* local variable definition */
   int a = 10;

   /* do loop execution */
   do
   {
      if( a == 15)
      {
         /* skip the iteration */
         a = a + 1;
         continue;
      }
      printf("value of a: %d\n", a);
      a++;

   }while( a < 20 );

   return 0;
}
```

Comando de seleção: **switch**

- O comando switch permite selecionar uma entre várias alternativas. Embora construções if-else possam executar testes para a escolha de uma entre várias alternativas, o comando switch é mais limpo.
- **Sintaxe**: {
- **switch** (variável ou constante)
- **case** constante1:
  - Instrução;
  - Instrução;
  - break;
- **case** constante2:
  - Instrução;
  - Instrução;
  - break;
- **default**:
  - Instrução;
- }

Comando de seleção: **switch**

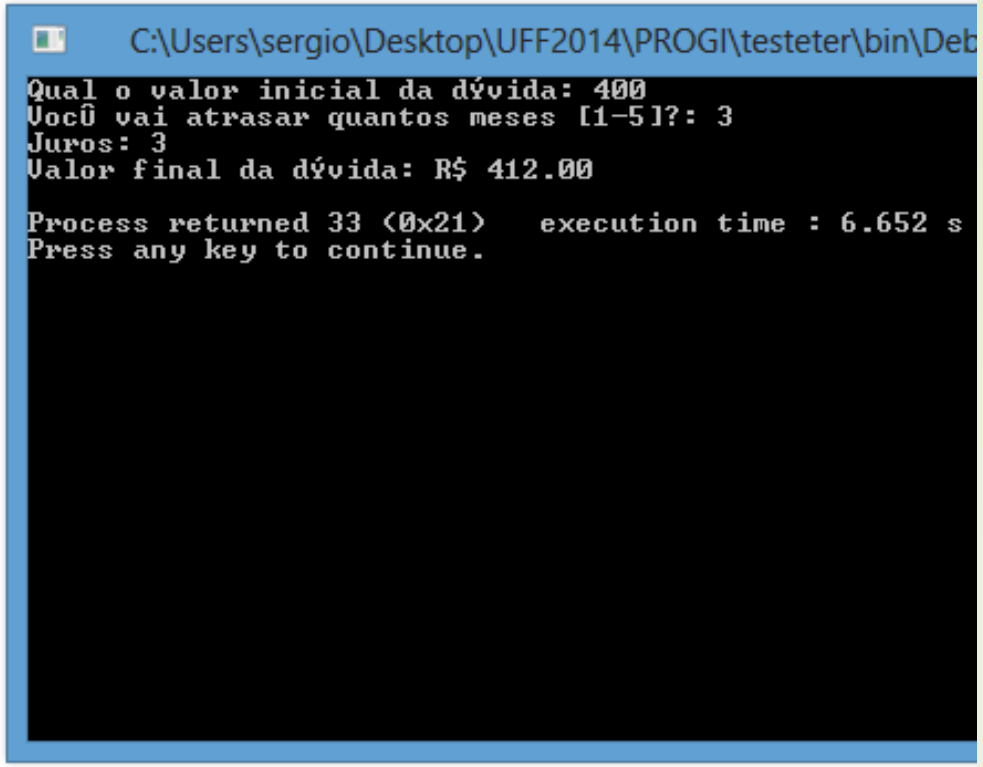
- O corpo de cada comando switch é composto por um número qualquer de instruções e a última é uma instrução **break**, o que causa uma saída imediata do corpo do switch.

- Se a instrução **break não estiver presente**, todas as instruções, a partir do caso escolhido até o término serão executadas, mesmo sendo pertencentes aos casos seguintes.

```c
#include <stdio.h>
int main ()
{
    int num;
    printf ("Digite um numero: ");
    scanf ("%d",&num);
    switch (num)
        {
            case 9:
                    printf ("\n\nO numero e igual a 9.\n");
            break;
            case 10:
                    printf ("\n\nO numero e igual a 10.\n");
            break;
            case 11:
                    printf ("\n\nO numero e igual a 11.\n");
            break;
            default:
                    printf ("\n\nO numero nao e nem 9 nem 10 nem 11.\n");
        }
    return(0);
}
```

```c
#include <stdio.h>

int main(void)
{
        int continuar=1;
        char opcao;

        while(continuar)
        {
            system("clear || cls");
            printf("Repetir? (S/s)im (N/n)ao\n");
            scanf(" %c",&opcao);

            switch(opcao)
            {
                case 's':
                case 'S':
                    break;

                case 'n':
                case 'N':
                    continuar = 0;
            }

        }

}
```

```c
#include <stdio.h>

int main(void)
{
        float valor_i,
              valor_f;
        int   juros=0;

        int meses;

        printf("Qual o valor inicial da dívida: ");
        scanf("%f", &valor_i);

        printf("Você vai atrasar quantos meses [1-5]?: ");
        scanf("%d", &meses);

        switch( meses )
        {
            case 5:
                juros++;
            case 4:
                juros++;
            case 3:
                juros++;
            case 2:
                juros++;
            case 1:
                juros++;
                break;
            default:
                printf("Você não digitou um valor válido de meses\n");

        }
        printf("Juros: %d%\n",juros);
        valor_f=( (1 + (juros/100.0))*valor_i);
        printf("Valor final da dívida: R$ %.2f\n", valor_f);

}
```



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

int main () {
int n1, n2;
char y, resp;

do{
    printf ("Operação Desejada");
    scanf ("%c",&y);
    printf ("Primeiro valor");
    scanf ("%d",&n1);
    printf ("Segundo valor");
    scanf ("%d",&n2);
    switch(y){
    case'+':
        printf ("Result.:%d", n1+n2);
    case'-':
        printf ("Result.:%d", n1-n2);
    case'*':
        printf ("Result.:%d", n1*n2);
    case'/':
        printf ("Result.:%d", n1/n2);
    default:
        printf ("Operação Inválida!");
    }
    printf("Deseja Realizar outra operação?");
    scanf("%c", &resp);
}while (resp=='S' || resp=='s');
getch();
}

```

Funções

- Uma função é um conjunto de instruções desenhadas para cumprir uma tarefa particular.
- Qualquer sequencia de instruções que apareçam repetidas em vários pontos e um programa são ótimas candidatas a serem transformadas em uma função.
- Uma função pode ser utilizada inúmeras vezes no decorrer de um programa.

Funções definidas pelo usuário

Formato Geral de uma Função em C

tipo_da_funcao **NomeDaFuncao** (Lista_de_**Parametros**)

{

// corpo da função

return x;

}

A **Lista_de_Parâmetros**, também é chamada de **Lista_de_Argumentos**, é opcional.

Toda função retorna um valor. Este valor tem um formato e é definido na própria confecção da função.

Protótipo de uma função

- Uma função **NÃO** pode ser chamada sem antes ter sido declarada!
- A declaração de uma função é dita protótipo da função. É geralmente colocada no início do programa onde **deixa evidente** o **tipo** da função e os **argumentos** que ela irá utilizar.
- O protótipo da função permite que o compilador verifique a sintaxe de chamada da mesma.
- O **tipo** de uma função é determinado pelo **tipo do dados que ela retornar** e **NÃO** pelo tipo de argumentos que ela recebe!

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:

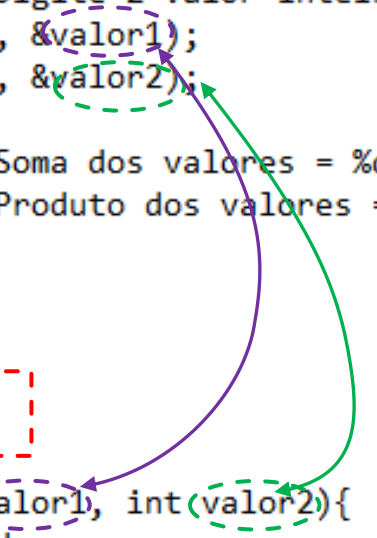
- As variáveis valem no bloco que são definidas;
- as variáveis definidas dentro de uma função recebem o nome de variáveis locais;
- os parâmetros formais de uma função valem também somente dentro da função;
- **uma variável definida dentro de uma função não é acessível em outras funções, MESMO ESTAS VARIÁVEIS TENHAM NOME IDÊNTICOS.**

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

int soma(int valor1, int valor2);  // PROTOTIPO da FUNCAO

int valor1, valor2, resp;

int main()
{
    printf("\nDigite 2 valor inteiros:");
    scanf("%d", &valor1);
    scanf("%d", &valor2);

    resp= soma(valor1, valor2);    // Variavel recebe o retorno da funçao SOMA

    printf("Soma dos valores = %d", resp );

    return 0;
}

// Funcao SOMA

int soma(int valor1, int valor2){
   int resultado;
   resultado = valor1 + valor2;
   return resultado;
}
```

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

int soma(int valor1, int valor2);  // PROTOTIPO da FUNCAO

int valor1, valor2;

int main()
{
    printf("\nDigite 2 valor inteiros:");
    scanf("%d", &valor1);
    scanf("%d", &valor2);
    printf("Soma dos valores = %d", soma(valor1, valor2));

    return 0;
}

// Funcao SOMA. recebe dois inteiros, soma os valores e devolve o resultado
int soma(int valor1, int valor2){
   int resultado;
   resultado = valor1 + valor2;
   return resultado;
}
```

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

float media(float nota1, float nota2, float nota3);   / prototipo da funcao

int main()
{
    float n1, n2, n3;
    printf("Digite 3 notas de provas\n");
    scanf("%f %f %f", &n1, &n2, &n3);

    printf("\nMedia do aluno= % 3.2f", media(n1,n2,n3));
    return 0;
}

// Funcao MEDIA

float media( float nota1, float nota2, float nota3){

return ((nota1+nota2+nota3)/3);  // Aqui eu retorno uma expressao

}
```

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

int soma(int valor1, int valor2);  // PROTOTIPO da FUNCAO
int produto( int v1, int v2);      // PROTOTIPO da FUNCAO

int valor1, valor2;

int main()
{
    printf("\nDigite 2 valor inteiros:");
    scanf("%d", &valor1);
    scanf("%d", &valor2);

    printf("\nSoma dos valores = %d", soma(valor1, valor2) );
    printf("\nProduto dos valores = %d", produto(valor1, valor2) );

    return 0;
}

// Funcao SOMA

int soma(int valor1, int valor2){
   int resultado;
   resultado = valor1 + valor2;
   return resultado;
}

// funcao PRODUTO

int produto(int v1, int v2){
   return v1*v2;
}
```



Exercícios- use funções para:

(1) Faça um programa que leia um número, passe este número como parâmetro para uma função que retorne 0 se o número for par ou 1 se o número for ímpar.

(2) Escreva uma função que receba como argumento o ano e retorne 1 se for ano bissexto ou 0 se não for um ano bissexto.

(3) Faça uma função que receba como argumento os valores dos lados de um triângulo, a função deverá retornar 0 se triângulo for equilátero, 1 se for isósceles ou 2 se for escaleno.

(4) Faça um programa que peça um número para calcular o fatorial. O programa deve ter uma função que verifique se a entrada é valida (maior que 0)

(5) Escreva um programa que use uma função que calcule as seguintes operações entre dois números dados: Adição, Diferença, Multiplicação, Divisão, Exponenciação e Media.

P1 vai ate funções!