UNIVERSIDADE FEDERAL DE MINAS GERAIS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
DCC003 – AEDS 1 – profa. Mirella Moro

PARTE 1 - INTRODUÇÃO AO C

Estrutura básica do programa:

```
diretivas para o pré-processador
declaração de variáveis globais
main () {
        declaração de variáveis locais da função main
        comandos da função main
}
```

1. DIRETIVAS PARA O PRÉ-PROCESSADOR

#include <stdio.h>	Funções de entrada e saída
#include <stdlib.h>	Funções padrão
#include <math.h>	Funções matemáticas
#include <string.h>	Funções de texto
#include <system.h>	Funções do sistema

2. DECLARAÇÃO DE VARIÁVEIS

Declaram as variáveis e seus tipos
Os nomes das variáveis devem conter apenas letras, dígitos e o símbolo _
Os principais tipos são: int, float, double e char
Exemplos de declaração de variáveis (sempre terminam com ;)

```
int n;
int quantidade_valores;
float x, y, somaValores;
char sexo;
char nome[40];
```

NOTE QUE C diferencia letras maiúsculas de minúsculas!
int n, N; → n é diferente de N!

3. ATRIBUIÇÃO

Atribui o valor da direita à variável da esquerda
O valor pode ser uma constante, uma variável ou uma expressão
Exemplos de atribuição de valores a variáveis:

```
x = 4;          → lemos x recebe 4
y = x + 2;      → tipo numérico (int, double, float)
y = y  + 4;
valor = 2.5;    → double ou float
sexo = 'F';     → char
```

4. ENTRADA DE VALORES VIA TECLADO

Utilizar a função scanf
scanf ("formatos", &var1, &var2,...)
Exemplo de utilização da função scanf:

```
int i, j;    float x;    char c;
scanf("%d", &i);
scanf("%d %f", &j, &x);
scanf("%c", &c);
scanf("%s", nome);
```

FORMATO:

%d inteiro	%f float
%lf double	%c char
%s palavra	

5. SAÍDA DE VALORES NA TELA

Utilizar a função printf
printf ("formatos", var1, var2,...)
Exemplos:

```
int i, j;    float x;    char c;
printf("%d", i);
printf("%d, %f", j, x);
printf("%c", c);
printf("%s", nome);
```

6. OPERADORES MATEMÁTICOS

```
+ soma          - subtração
* multiplicação  / divisão
% resto (também chamado módulo)
```

Exemplos:

```
x = x + y;
z = x/y;
```

OPERAÇÕES SIMPLIFICADAS

Vamos utilizar muitas vezes a soma de 1 ou a subtração de 1, os quais podem ser simplificados da seguinte maneira:

```
x = x + 1; é a mesma coisa que x++;
y = y - 1; é a mesma coisa que y--;
```

Outras operações podem ser simplificadas:

```
x *= 2; é a mesma coisa que x=x*2;
x -= 5; é a mesma coisa que x=x-5;
```

7. EXEMPLO COMPLETO

```
#include <stdio.h>
#include <stdlib.h>
main()
{
  double n1, n2, n3, media;
  scanf ("%lf %lf %lf",&n1, &n2, &n3);
  media=(n1+n2+n3)/3;
  printf ("%lf",media);
  system("PAUSE");
}
```

8. DETALHES DE PROGRAMAÇÃO

Termine todas as linhas com ;

Sempre salve o programa antes de compilar

Sempre compile o programa antes de executar

Quando ocorrer um erro de compilação, dê um duplo clique sobre a mensagem de erro para destacar o comando errado no programa

Verifique também a linha anterior, que pode ser a responsável pelo erro, especialmente se faltar o ;

Use comentários, iniciados por //

PARTE 2 – COMANDOS DE DECISÃO

Comandos de decisão ou desvio definem estruturas de algoritmos que não são totalmente sequenciais. Com as instruções de desvio pode-se fazer com que o algoritmo proceda de uma ou outra maneira, de acordo com as decisões lógicas tomadas em função dos dados ou resultados anteriores. Em C, essas estruturas são:

```
if (condição) comando;
if (condição) comando; else comando;
switch (valor) {case comando; ... }
```

1. OPERADORES LÓGICOS

&& (E lógico) :	if(a>2 && b<3)
|| (OU lógico):	if(a>1 || b<2)
! (NÃO lógico):	if(!var)

2. SE ENTÃO SENÃO

```
if (condição) comando;
else comando;
```

OU

```
if (condição) {
        comando1;
        comando2;  }
else  {
        comando3;
        comando4;  }
```

3. CASO ENTÃO

```
switch (numero)
{
  case 1: printf ("Janeiro\n"); break;
  case 2: printf ("Fevereiro\n");break;
  case 3: printf ("Marco\n"); break;
     ...
 default: printf (“Mes invalido\n");
}
```

PARTE 3 – COMANDOS DE REPETIÇÃO

Até agora escrevemos todas as ações, mesmo repetições de ações praticamente iguais. Por exemplo, no algoritmo para o cálculo da média de quatro números, líamos 4 vezes, 4 valores para dentro de 4 variáveis. Mas também poderíamos:

- Ler um valor para uma variável e repetir isso 4 vezes, adicionando cada valor lido ao total em uma outra variável, a cada repetição
- Após as 4 repetições, a soma dos 4 números estaria acumulada na outra variável, bastando uma instrução para dividi-la por 4 e assim obter a média

PSEUDO-CÓDIGO

```
real: n, soma, media
inteiro:i
soma=0
i=0
repita
   ler n
   soma = soma + n
   i = i+1
enquanto i < 3
media = soma/i
exibir media
```

Se uma ação se repete em um algoritmo, em vez de escrevê-la várias vezes, em certos casos podemos resumir anotando uma vez só e solicitando que ela se repita, usando uma das *estruturas de repetição*. Podemos pedir que uma ação (ou um conjunto de ações) seja executada um número definido ou indefinido de vezes, ou enquanto um estado permanecer ou até que um estado seja atingido.

As principais estruturas de repetição são:

```
Enquanto (condição)  repetir
      instrução
Repetir instrução
      enquanto (condição)
Para variável = vInicio até  vFim
repetir instrução
```

Em C:

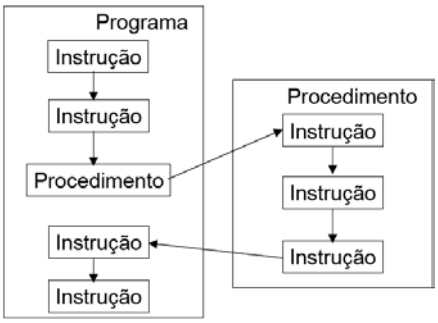
```
while (condição)
   comando;

do {  comando;
   } while (condição);

for (var=inicio; condição; incremento)
  comando;
```

PARTE 4 – FUNÇÕES, PONTEIROS, STRINGS

Além das operações matemáticas básicas, as linguagens possuem funções pré-programadas que retornam valores de funções comuns da matemática. Funções pré-programadas ou internas fazem parte do repertório de recursos disponíveis nas bibliotecas das linguagens de programação. O programador pode desenvolver novas funções para atender ao seu projeto.

1. MODULARIZAÇÃO

Subdividir um problema grande e complexo em problemas menores. Desenvolvimento e implementação usando subprogramas (subrotinas, funções, etc.). Construção de bibliotecas para reutilização dos subprogramas. Disciplina de desenvolvimento de programas:



2. FUNÇÕES

- Retorna um valor (ou é declarada como void)
- Recebe um ou mais como parâmetros
- Precisa de uma assinatura (ou cabeçalho):
  1. Determina o tipo do valor retornado
  2. Determina o(s) tipo(s) do(s) parâmetro(s)

float f = pow (f, 2); ➔ f = f^2

assinatura = float pow (float x, float y);

As funções em C são escritas da seguinte maneira:

```
tipo NomeFuncao (listaParâmetros) {
        comandos;
        return valor_retornado; }
```

Exemplo:

```
int SomaDoisNumeros(int A, int B)
{ return A +B; }
double pot (double A, double B) {
      double ret=1;
      for (int i =0; i <B; i++)
         ret*=A;
      return ret; }
void ImprimeTexto()
{     printf(“Ola Mundo”); }
main() {
      int r = SomaDoisNumeros (30, 50);
      double p = pot(2.0,100);
      ImprimeTexto(); }
```

3. FUNÇÕES SEM PARÂMETRO

E se a função não retorna um valor?

E se ela não tem um parâmetro?

Void é um termo que indica ausência

Funções que não retornam valores em seus nomes são declaradas como tipo *void*

Funções que não utilizam parâmetros são declaradas com parâmetro *void*

Exemplo: Para escrever linhas de ***

```
void apresente_linha (void)  {
  for (int i=1;i<20;i++) printf("*");
    printf("\n");    }
```

4. PASSAGEM DE PARÂMETRO

(1) Passagem por Valor: o parâmetro passado por valor é copiado, ou seja, o valor da variável utilizada como parâmetro não é alterado.

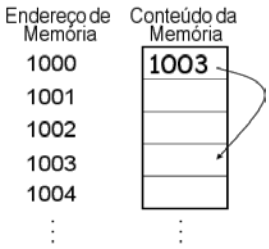
(2) Passagem por Referência: O endereço do parâmetro passado por referência é atribuído à um ponteiro, ou seja, qualquer alteração no conteúdo apontado será refletida no conteúdo da variável utilizada como parâmetro

Exemplo:

```
void FuncaoQualquer(int A, int* B) {
        A = 1;
        *B= 2;
}
main()
{       int X = 0, Y = 0;
        FuncaoQualquer (X, &Y);
        printf("%d %d", X, Y);
        system("pause");    }
```

5. PONTEIROS

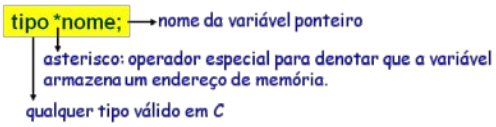
Toda e qualquer variável utilizada por um programa reside em determinado endereço de memória. O acesso ao endereço de uma variável pode ser feito simbolicamente através de seu nome. Essa referência é simbólica porque o endereço de memória propriamente dito NÃO é especificado.



ponteiros apontam !!!!

Por exemplo, a posição 1000 guarda um ponteiro para a posição 1003

Declaração de ponteiros:



&: operador unário que devolve o endereço de memória de seu operando.

Exemplo 1

```
int count, q, *m;
m = &count; //coloca em m o endereço da variável count
```

Lê-se: "m recebe o endereço de count"

Se a variável count está armazenada na posição de memória 2000, o valor de m será 2000.

Assim como qualquer variável, um ponteiro pode ser usado no lado direito de um comando de atribuição, para passar seu valor para um outro ponteiro.

Exemplo 2

```
int x, *p1, *p2;
p1 = &x; //p1 aponta para x
p2 = p1; //p2 recebe p1 e também passa a apontar para x
printf("%p", p2); //escreve o endereço de x
```

Exemplo 3

```
int x=10, *p1, *p2; // x é variável tipo inteiro, p1 e p2 são ponteiros para variáveis tipo inteiro
p1 = &x; //p1 recebe o endereço de x (→ x)
p2 = p1; //p2 recebe p1, também →
printf("%d", *p2); //escreve o valor de x
printf("%d", p2); //escreve lixo
printf("%p", p2); //escreve o endereço de x
```

6. STRINGS

Uma constante do tipo String é uma cadeia de caracteres entre aspas: "Isso é uma constante string"

Uma variável capaz de armazenar um String é na verdade variável composta do tipo char:

```
char nome[20];
nome = "Jose da Silva";
```

→ Armazenamento na memória

```
char ch[8] = "lagarto";
printf ("%s\n", ch);
gets(ch);
scanf ("%s", &ch);
#include <string.h>
int comp = strcmp (s1, s2);
int tamanho = strlen(ch);
strcpy (destino, origem);
strcat (s1, s2);
s1 = strupr(s1);
s2 = strlwr(s2);
```

EXERCÍCIOS DE REVISÃO

DECISÃO

1) Dados dois números A e B, some 100 ao maior número e mostre na tela.

2) Escreva um algoritmo para determinar se uma pessoa é maior ou menor de idade.

3) Faça um algoritmo que leia a quantidade comprada de um produto e seu preço.

- Se o preço total a ser pago for inferior a 100, então forneça um desconto de 5%.
- Se o preço total a ser pago ficar entre 100 e 1000, então forneça um desconto de 10%.
- Se o preço total a ser pago for superior a 1000, então forneça um desconto de 15% e escreva na tela que o cliente será cadastrado como 'Cliente Vip'.

REPETIÇÃO

1) Construir um algoritmo que calcule a média aritmética de vários valores inteiros positivos, lidos externamente. O final da leitura acontecerá quando for lido um valor negativo.

2) Escreva um algoritmo que calcule a média aritmética das 3 notas dos alunos de uma classe. O algoritmo deverá ler, além das notas, o código do aluno e deverá ser encerrado quando o código for igual a zero.