Paradigmas de Programação

Josino Rodrigues

Introdução

- **Características do C**
  - Total interação com o sistema operacional.
  - Código compacto e rápido, quando comparado ao código de outras linguagens.

Programando em C

Programando em C

- **Estrutura de um Programa em C**
  - Inclusão de bibliotecas (se necessário)
  - Inclusão das constantes (se necessário)
  - Comentários (se desejar)
  - Programa principal (obrigatório)
    - Declaração de variáveis (se necessário)
    - Seqüência de comandos

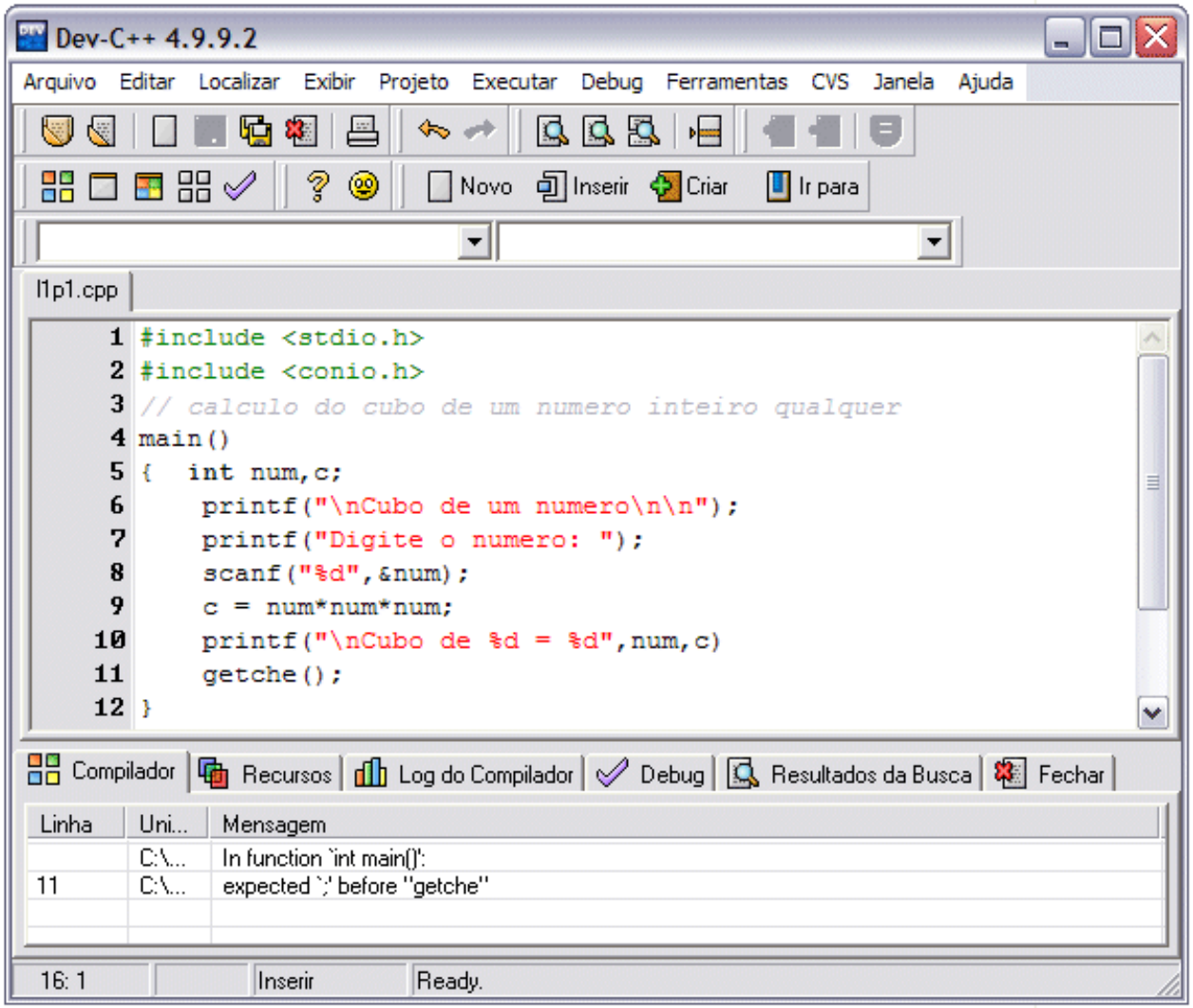
Programando em C

- **Estrutura de um Programa em C**

```
#include <stdio.h>
#include <conio.h>
/* Calcular o volume de uma esfera de raio R,
   onde R será fornecido pelo usuário. */
main () {
        float v, r;
        printf("Volume de uma Esfera\n \n");
        printf("\nRaio da Esfera:");
        scanf("%f",&r);
        v = (4*3.1415*r*r*r)/3;
        printf("\n\nVolume de uma Esfera = %.2f", v);
        getche();
}
```

Programando em C

- **Conhecendo o Dev-C++**



Programando em C

- Tipos de Dados Primitivos
  - Dados são representados pelas informações a serem tratadas (processadas) por um computador.
    - **Inteiro**
    - **Real**
    - **Caractere**
    - **Lógico**

Programando em C

- Tipos de Dados Primitivos
  - **Inteiro**
    - Toda e qualquer informação numérica que pertença ao conjunto dos números inteiros relativos (negativa, nula ou positiva).

Tipo	Tamanho	Valores
short	16 bits	-32.768 a +32.767
int	32 bits	-2.147.483.648 a + 2.147.483.647
long	64 bits	-9.223.372.036.854.775.808 a +9.223.372.036.854.775.807

Programando em C

- Tipos de Dados Primitivos
  - **Real**
    - Toda e qualquer informação numérica que pertença ao conjunto de números reais (negativa, positiva ou nula).

Tipo	Tamanho	Valores
float	32 bits	3.4E-38 a 3.4E+38
double	64 bits	1.7E-308 a 1.7E+308

Programando em C

- Tipos de Dados Primitivos
  - **Caractere**
    - Toda e qualquer informação composta por um caractere alfa numéricos (a, b, c,...z, A, B, C,...Z, 0...9) e /ou especiais (ex; #?@!<?).
    - O caractere deve ser indicado entre apóstrofo ‘ ’
      - **`char`**

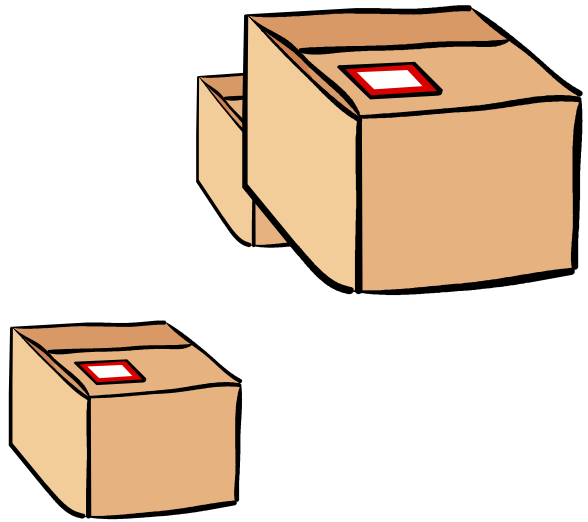
Programando em C

- **Identificadores**
  - Representam os nomes escolhidos para rotular as variáveis, constantes, procedimentos e funções.
  - Regras:
    - Primeiro caractere deve ser uma letra;
    - Os nomes devem ser formados por caracteres pertencentes ao seguinte conjunto : {a,b,c,..z, A,B,C,...Z, 0,1,2,...,9,_};
    - Não deve haver espaço em branco;
    - Não deve haver identificadores repetidos;
    - Existe distinção de maiúsculas e minúsculas;
    - Os nomes escolhidos devem ser claros a fim de explicitar seu conteúdo uso, mas
    - também não deve ser extenso para não dificultar a escrita.

Programando em C

- **Variáveis**
  - Unidades básicas de armazenamento das informações em programação.
  - As variáveis representam espaços onde podemos armazenar e manipular dados.
- **Declaração de Variáveis**
  - Precisamos declarar as variáveis para informar quais dados serão manipulados pelo programa.
  - A declaração de variáveis deve seguir a seguinte sintaxe:
    - **Tipo Lista de Variáveis;**
    - **Exemplo:**

```
int x;
float y, z;
char letra;
char nome[20];
```



Programando em C

- **Constantes**
  - Diferente das variáveis, as constantes mantêm seu valor ao longo do programa.
  - Para indicar ao compilador que se trata de uma constante, a diretiva #define é utilizada.
  - Sintaxe: #define <identificador> <valor>
  - Exemplos:

```
#define pi 3.1416
#define dias 7
#define disciplina "Matemática"
#define letra 'a'
```

Programando em C

- **Operações Básicas**
  - Os operadores são o meio pelo qual incrementamos, decrementamos, comparamos e avaliamos dados dentro do computador.
  - Temos quatro tipos de operadores:
    - Operador de Atribuição
    - Operadores Aritméticos
    - Operadores Relacionais
    - Operadores Lógicos

Programando em C

- **Operador de Atribuição**
  - *Sintaxe: NomeDaVariavel* = Valor, Variável ou Expressão;
  - Esse operador armazena o Valor, Variável ou Expressão na variável. É importante que o tipo do Valor, Variável ou Expressão seja compatível com o tipo da variável.
  - Exemplo:

```
MEDIA = 7;

MEDIA = X;

MEDIA = X + Y;
```

Programando em C

- **Operadores Aritméticos**
  - Esses operadores serão utilizados em expressões para realizar operações aritméticas com variáveis.

Operador	Descrição
+	Adição
-	Subtração
*	Multiplicação
/	Divisão Inteira
%	Retorna o resto da **divisão inteira** de **a** por **b**

Programando em C

- **Operadores Aritméticos**
  - **Exemplo: Suponha que A = 6, B = 1, C = 16 e D = 6.5, então:**

```
A = A + 1;    =>   A = 7
B = B * C;    =>   B = 16
C = C – A;    =>   C = 9
A = C % A;    =>   A = 2
A = C/A;      =>   A = 4
D = D/2;      =>   A = 3.25
```

Programando em C

- **Operadores Aritméticos**
  - **Prioridades**

1º	Operações embutidas em parênteses "mais internos"
2º	Resto
3º	Multiplicação e/ou divisão
4º	Adição e/ou Subtração

```
A = B + C * C => A = 1 + 16 * 16 => A = 1 + 256 => A = 257
A = (B + C) * C => A = (1 + 16) * 16 => A = 17 * 16 => A = 272
```

Programando em C

- **Principais funções da biblioteca math.h**
  - **Funções trigonométricas**
    - `sin (double);` → `y = sin(x);`
    - `cos (double);` → `y = cos(x);`
    - `tan (double);` → `y = tan(x);`
  - **Exponenciais e logaritmos**
    - `exp (double);` → `y = exp(x);`
    - `log (double);` → `y = log(x);`
    - `log10 (double);` → `y = log10(x);`

Programando em C

- **Principais funções da biblioteca math.h**
  - **Raiz e potência**
    - **`sqrt(double);`**
      - `y = sqrt(x);`
      - `y = sqrt(16);`
    - **`pow(double, double);`**
      - `p = pow(x, y);`
      - `p = pow(x,4);`
      - `p = pow (3,y);`

Programando em C

- **Principais funções da biblioteca math.h**
  - **Arredondamentos**
    - `floor(double);`
      - Retorna o primeiro número inteiro menor que o número que desejamos arredondar. Exemplo, se x = 1.09 será retornado 1.
      - `i = floor(x);`
    - `ceil(double);`
      - Retorna o primeiro número inteiro maior que o número que desejamos arredondar. Exemplo, se x = 1.09 será retornado 2.
      - `i = ceil(x);`

Programando em C

- **Atribuições Reduzidas**
  - C oferece operadores de atribuição que são redução de outros operadores.

    `+=`

    `-=`

    `/=`

    `*=`
  - **Exemplos**

    `x += 3; equivale a x = x + 3;`

    `y -= 5; equivale a y = y – 5;`

Programando em C

- **Atribuições pré e pós fixados**
  - `++`
  - `– –`
  - **`Exemplo:`**
    - `i++;`
    - `i--;`
    - `x = y + (i++);`
    - `--i;`

Expressão	Valor de i utilizado na avaliação	Valor da expressão	Valor final de i
5 + (i++)	5	10	6
5 + (i--)	5	10	4
5+(++i)	6	12	7
5+(--i)	4	8	3

Programando em C

- **Conversão de tipo (Casting)**
  - Algumas vezes queremos, momentaneamente, modificar o tipo de dado representado por uma variável.
  - Por exemplo: declaramos uma variável como int e queremos, momentaneamente, que seu conteúdo seja apresentado como float.
  - **Sintaxe: `(`*`tipo`*`)`*`variável`**
    - onde tipo é o nome do tipo ao qual queremos converter o dado armazenado em variável.
  - **Exemplo:**

```
int num;
float valor;
valor = 13.0;
num = (int)valor % 2;
```

Programando em C

- **Comando de Entrada de Dados**
  - A execução da instrução de leitura pressupõe que os dados serão fornecidos via teclado e serão armazenados na memória (variáveis).

- Sintaxe: **`scanf("string de controle", lista de argumentos);`**
  - **string de controle:**
    - %d - leitura de números inteiros
    - %f - leitura de números reais
    - %c - leitura de caractere
    - %s - leitura de cadeia de caracteres

  - **Lista de argumentos:** variáveis onde os valores digitados via teclado deverão ser armazenados. A variável deve ser precedida do &.

- **Exemplos:**

```
scanf("%d",&idade);
scanf("%f",&nota);
```

Programando em C

- **Comando de Entrada de Dados**
  - **Atenção:** cadeias de caracteres devem ser lidas utilizando o comando gets. Não deve ser usado o scanf! Antes da leitura, deve ser utilizado o comando que limpa o buffer de entrada.
  - **Exemplo:**

```
char nome[20];
fflush(stdin);
gets(nome);

char letra;
fflush(stdin);
scanf("%c",&letra);
```

Programando em C

- **Comando de Saída de Dados**
  - A execução do comando de saída fará com que informações sejam mostradas na tela.
  - **Sintaxe: printf("string de controle", lista de argumentos);**
    - **Códigos especiais:**
      - **\n avanço de linha**
      - **\t tabulação (tab)**
      - **\" aspas duplas**
      - **\\ barra**
      - **\a Alerta sonoro**

Programando em C

- **Comando de Saída de Dados**
  - Necessariamente você precisará ter tantos argumentos quantos forem os comandos de formatação na “string de controle”. Se isto não ocorrer, a tela exibirá sujeira ou não exibirá qualquer dado.
    - **Códigos de impressão formatada:**
      - **%c caractere**
      - **%d inteiro**
      - **%f real**
      - **%s string**
      - **%% o caractere '%'**

Programando em C

- **Comando de Saída de Dados**
  - **Exemplos:**
    - `printf("Uma linha");`
    - `printf("Uma linha\nDuas linhas");`
    - `printf("Os números são: %d e %d\n",7,8);`
    - `printf("Caruaru está a %f Km de Recife",dist);`
    - `printf("%d%% de %d = %f\n", p, x, x*(p/100.0));`

Programando em C

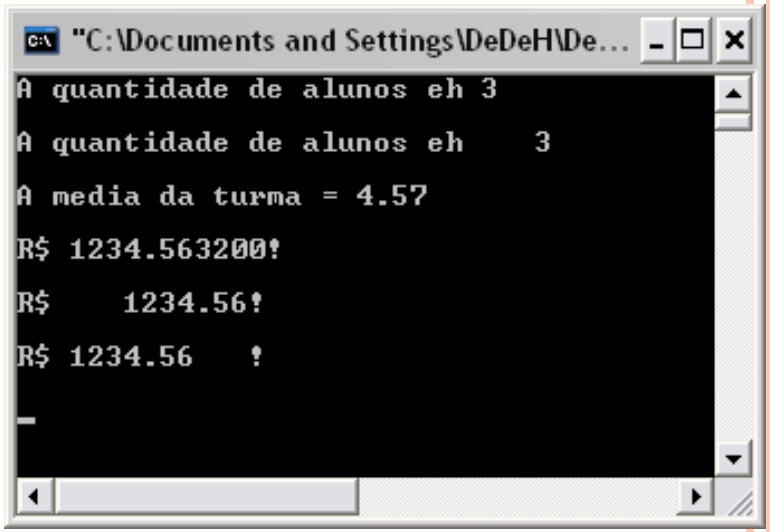
- **Comando de Saída de Dados**
  - **Formatação:** o tamanho de campos de impressão é indicado logo após o '%' e antes do tipo do campo.
  - **Exemplos:**
    - `printf("A quantidade de alunos eh %d!\n",quant);`
    - `printf("A quantidade de alunos eh %4d!\n",quant);`
    - `printf("A media da turma = %.2f\n",media);`
    - `printf("R$ %.2f!\n",1234.5632);`
    - `printf("R$ %10.2f!\n",1234.5632);`
    - `printf("R$ %-10.2f!\n",1234.5632);`

Programando em C

- **Comando de Saída de Dados**

```
#include <stdio.h>
#include <conio.h>

main ()
{int quant = 3;
 float media = 4.5678;
printf("A quantidade de alunos eh %d\n\n", quant);
printf("A quantidade de alunos eh %4d\n\n", quant);
printf("A media da turma = %.2f\n\n", media);
printf("R$ %2f!\n\n", 1234.5632);
printf("R$ %10.2f!\n\n", 1234.5632);
printf("R$ %-10.2f!\n\n", 1234.5632);
getche();
}
```

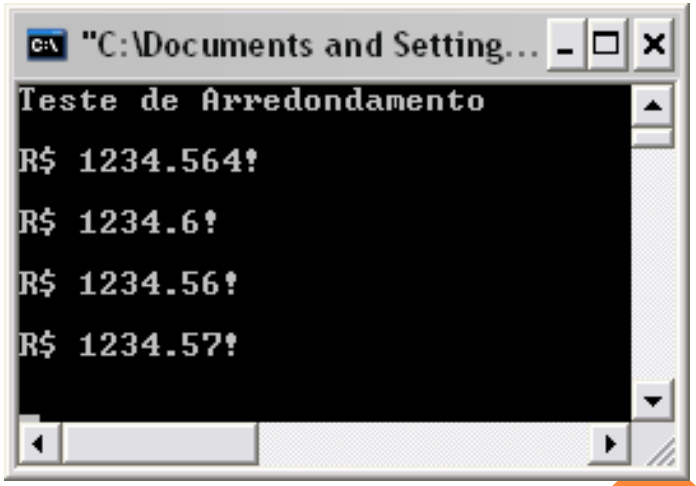


Programando em C

- **Comando de Saída de Dados**

```
#include <stdio.h>
#include <conio.h>

main ()
{
printf("Teste de Arredondamento\n\n");
printf("R$ %.3f!\n\n", 1234.5639);
printf("R$ %.1f!\n\n", 1234.5632);
printf("R$ %.2f!\n\n", 1234.5632);
printf("R$ %.1f!\n\n", 1234.5652);
getche();
}
```



Programando em C

- **Palavras Reservadas**

auto	default	float	unsigned
break	do	for	return
case	double	goto	short
char	else	union	signed
const	typedef	if	sizeof
switch	while	int	static
volatile	enum	long	struct
continue	extern	register	void

Programando em C

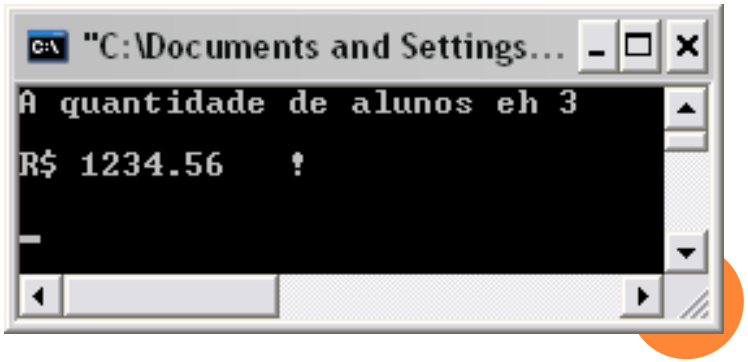
- **Comentários**
  - Os comentários são utilizados para documentar um programa, bem como podem ser úteis para encontrar erros no código.
  - Comentários podem ser colocados em qualquer lugar de seu programa.
  - **Comentário de Bloco**
    - Comentários começam com barra-asterisco (/*) e terminam por asterisco-barra (*/).

```
/*************************************************
 * Programa: corrida.c
 * Programador: Antonio Oliveira e Verlaynne Rocha
 * Data: 15/01/2011
 * Programa para imprimir o resultado de uma corrida
 *************************************************/
```

Programando em C

- **Comentários**
  - **Comentário de Bloco**

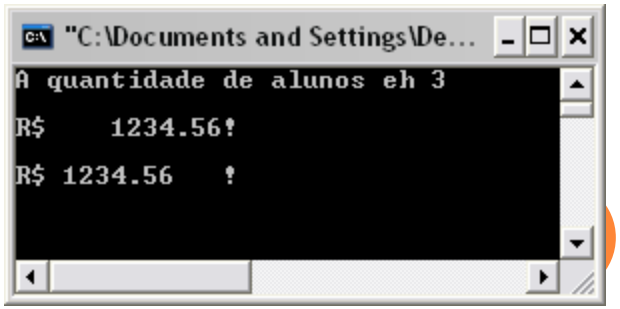
```
#include <stdio.h>
#include <conio.h>
/* ********************************
* Programa exemplo de comentarios *
*Programador: Antonio Oliveira e Verlaynne Rocha ****
* Data: 15/01/2011
**************************************/
main ()
{int quant = 3;
 float media = 4.5678;
printf("A quantidade de alunos eh %d\n\n", quant);
/*printf("A quantidade de alunos eh %4d\n\n", quant);
printf("A media da turma = %.2f\n\n", media);
printf("R$ %2f!\n\n", 1234.5632);
printf("R$ %10.2f!\n\n", 1234.5632);*/
printf("R$ %-10.2f!\n\n", 1234.5632);
getche();
}
```



Programando em C

- **Comentários**
  - **Comentário de Linha**
    - o comentário é feito colocando-se // no início do texto de comentário.
    - Neste caso não tem um símbolo para fechar o comentário.

```
#include <stdio.h>
#include <conio.h>
// Programadores Antonio Oliveira e Verlaynne Rocha
main ()
{int quant = 3;
 float media = 4.5678;
printf("A quantidade de alunos eh %d\n\n", quant);
//printf("A quantidade de alunos eh %4d\n\n", quant);
//printf("A media da turma = %.2f\n\n", media);
//printf("R$ %2f!\n\n", 1234.5632);
printf("R$ %10.2f!\n\n", 1234.5632);
printf("R$ %-10.2f!\n\n", 1234.5632);
getche();
}
```



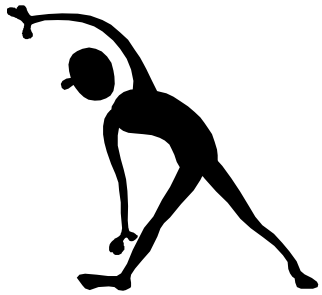
Programando em C

- **Exercícios**

1. Faça um programa que leia um valor em graus Fahrenheit e imprima o seu equivalente em graus Celsius.

$$C = (F - 32)*5/9$$

2. Faça um programa para ler o número de votos brancos, nulos e válidos. Calcular e escrever o percentual que cada um representa em relação ao total de eleitores.



Programando em C

- **Exercícios**

3. Dada as seguintes entradas: km inicial, km final, litros consumidos, preço do litro de combustível. Faça um programa que imprima o seguinte Relatório: distancia percorrida, Valor total gasto e Km por litro.

Programando em C

- **Operadores Relacionais**

Operador	Descrição
==	Igual
!=	Diferente (≠)
<	Menor
>	Maior
<=	Menor ou Igual (≤)
>=	Maior ou Igual (≥)

Programando em C

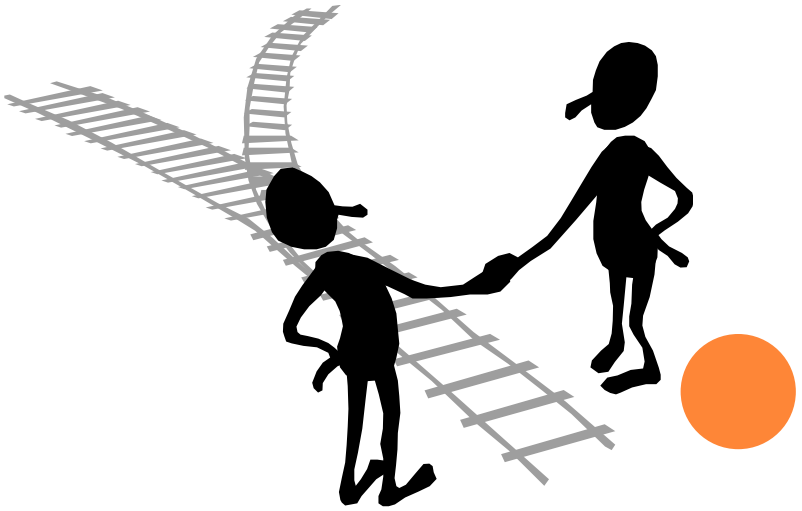
Operadores Lógicos

- Uma expressão lógica representa a união de operações relacionais permitindo que o resultado de várias expressões relacionais sejam transformado em um único resultado lógico.

Operadores Lógicos	Operador em C
E	&&
OU	||
NÃO	!

Programando em C

- **Comandos de Seleção (decisão ou desvio)**
  - Usado para tomar decisões, ou seja desviar a execução do programa de acordo com uma condição, podendo ser simples ou composta.
  - O comando de seleção é utilizado quando há a necessidade de avaliar certas possibilidades dos valores de variáveis e, de acordo com o resultado dessa análise, executar um conjunto específico de comandos.
  - **If...Else**
    - **Simples**
    - **Composto**
    - **Aninhado**
  - **Switch**



Programando em C

- **Comandos de Decisão (seleção ou desvio)**
  - **If...Else Simples**

Primeiro Caso: if com um único comando

if (Condição) Comando;	if (x==1) a = a + 3;

Segundo Caso: if com mais de um comando

if (Condição)
{ Comando;
...
Comando;
}
	if (x==1)
	{ a = a + 3;
	z = z * 2;
	h = 12/6;
	}

Programando em C

- **Comandos de Decisão (seleção ou desvio)**
  - **If...Else Composto**

Primeiro Caso: if e else com um único comando

```
if (Condição)
    Comando;
else
 Comando;
```

```
if (media>=7)
   printf("\nAprovado");
else
   printf("\nReprovado");
```

Segundo Caso: if com mais de um comando e else com um comando

```
if (Condição)
{   Comando;
          ...
   Comando;
}
else
Comando;
```

```
if (sal<260)
 {  bonus = sal * 0.1;
    nsal = sal + bônus;
}
else
   printf("\nsem bonus");
```

Programando em C

- **Comandos de Decisão (seleção ou desvio)**
  - **If...Else Composto**

Terceiro Caso: if com um comando e else com mais de um comando

```
if (Condição)
   Comando;
else
{  Comando;
      ...
   Comando;
}
```

```
if (sal >= 260)
   printf("\nsem bonus");
else
{  bonus = sal * 0.1;
   nsal = sal + bônus;
}
```

Programando em C

- **Comandos de Decisão (seleção ou desvio)**
  - **If...Else Composto**

Quarto Caso: if e else com mais de um comando

```
if (Condição)
{   Comando;
          ...
    Comando;
}
else
{   Comando;
          ...
    Comando;
}
```

```
if (sal >= 260)
{  bonus = sal * 0.2;
   nsal = sal + bônus;
}
else
{  bonus = sal * 0.4;
   sf = num_filho * 20;
   nsal = sal + bonus + sf;
}
```

Programando em C

- **Comandos de Decisão (seleção ou desvio)**
  - **If...Else Composto**

```
main()
{ float med;
  printf("Caderneta de Notas\n\n");
  printf("Media do Aluno: ");
  scanf("%f",&med);
  if (med>=7)
     printf("\n\nAluno aprovado com media
  %.1f\n",med);
  else
     printf("\n\nAluno reprovado com media
  %.1f\n",med);
  getche();
}
```

Programando em C

- **Comandos de Decisão (seleção ou desvio)**
  - **If...Else Aninhado**

```
#include <stdio.h>
#include <conio.h>

main (){
    int a,b;
    printf("Compara numeros\n\n");
    printf("A: ");
    scanf("%d", &a);
    printf("B:  ");
    scanf("%d", &b);
    if (a> b)
       printf("\n\n%d eh maior que %d\n", a,b);
    else{
        if (a< b)
           printf("\n\n%d eh menor que %d\n", a,b);
        else
        printf("\n\n A e B sao iguais:  %d\n", a);
    }
    getche();
}
```

Programando em C

- **Comandos de Decisão (seleção ou desvio)**
  - **Switch (Caso)**
    - Usado para tomadas de decisões quando a variável testada é um inteiro ou um único caractere.
    - Só serve para testar igualdade.
    - Mais restrito do que o if.

Programando em C

- **Comandos de Decisão (seleção ou desvio)**
  - **Switch (Caso)**

Caso simples

```
switch (variavel)
{  case valor :  Comando; ...break;
                  ...
   case valor: Comandos; ... break;
}
```

```
switch (cargo)
{  case 1: sal = sal + 200;  break;
   case 2: sal = sal + 400;  break;
   case 3: sal = sal + 600;  break;
}
```

Caso Composto

```
switch (variavel)
{  case valor :  Comando; ...break;
                  ...
   case valor: Comandos; ... break;
   default: comandos; ... break;
}
```

```
switch (cargo)
{  case 1: sal = sal + 200;  break;
   case 2: sal = sal + 400;  break;
   case 3: sal = sal + 600;  break;
   default : sal = sal + 100; break;
}
```

Programando em C

- **Comandos de Decisão (seleção ou desvio)**
  - **Switch com letras**

```
printf("\n\n\nSair (s-sim/n-nao)? ");
fflush(stdin);
scanf("%c",&letra);
switch (letra)
{ case 's': case 'S': printf("voce vai sair");
                      break;
  case 'n': case 'N': printf("voce vai continuar");
                      break;
  default: printf("escolha invalida");
           break;
}
```

Programando em C

- **Exercícios**

1. Faça um programa que leia um número inteiro e mostre uma mensagem indicando se este número é par ou ímpar, e se é positivo ou negativo.

2. Tendo como dados de entrada a altura e o sexo(1-feminino/2-masculino) de uma pessoa, construa um programa que calcula e escreve seu peso ideal, utilizando as seguintes fórmulas:

   para homens: (72.7*altura)-58

   para mulheres: (62.1*altura)-44.7

Programando em C

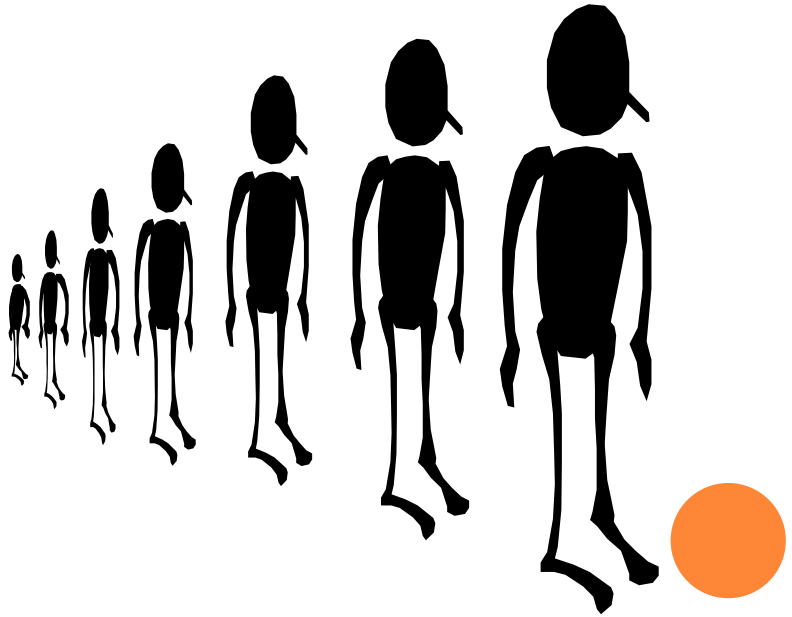
- **Exercícios**

3. Um hotel cobra R$ 50,00 reais a diária e mais uma taxa de serviços. A taxa de serviços é de:
2,50 por dia, se número de diárias <15
2,00 por dia, se número de diárias =15
1,50 por dia, se número de diárias >15

Faça um programa que ler a quantidade de dias que o hospede ficou no hotel e imprime a taxa e total a pagar.

Programando em C

- **Comandos de Repetição**
  - A estrutura de repetição permite que um bloco de instruções seja executado repetidamente uma quantidade controlada de vezes.
    - **For**
    - **While**
    - **Do/while**



Programando em C

- **Comandos de Repetição**
  - **For (Para)**

```
for (var-controle = valor-inicial; condição; incremento)
        Comando;
```

```
for (var-controle = valor-inicial; condição; incremento)
{       Comando;

        Comando;
}
```

```
for (var-controle = valor-inicial; condição; decremento)
        Comando;
```

Programando em C

- **Comandos de Repetição**
  - For (Para)
    - Exemplos

```
#include <conio.h>
#include <stdio.h>
main ()
{ int i;
  for (i=1; i<=10; i++)
      printf("\n %d", i);
  getche();
}
```

```
#include <conio.h>
#include <stdio.h>
main ()
{ int i;
  for (i=10; i>=1; i--)
      printf("\n %d", i);
  getche();
}
```

Programando em C

- **Comandos de Repetição**
  - **While (Enquanto – teste no início)**
    - enquanto a condição for verdadeira, o comando será executado repetidamente. Se a condição for falsa, então a repetição será interrompida.
    - A sintaxe do While é:

```
while (condição)
    comando;
```

```
while (condição)
{   comando;
    ...;
    comando;
}
```

Programando em C

- **Comandos de Repetição - While (Enquanto – teste no início)**

```
#include <stdio.h>
#include <conio.h>
// Calcula bonus do Cliente
main (){
    float valor ;
    int cod, cont;
    printf("Loja Compre Tudo\n\n");
    cont = 1;
    while (cont <= 5){
       printf("Codigo do Cliente %d: ", cont);
       scanf("%d", &cod);
       printf("Valor das Compras: ");
       scanf("%f", &valor);
       if (valor> 5000)
                printf("\n\n%Valor do bonus R$  %2.f\n", valor*0.1);
       else
                printf("\n\n%Valor do bonus R$  %2.f\n", valor*0.15);
       printf("\n\ntecle entre para continuar...\n");
       getche();
       cont++;
    }
}
```

Programando em C

- **Comandos de Repetição**
  - **Do/While (Faça...enquanto - teste no final)**
    - Repete um bloco de instruções enquanto a condição é verdadeira.
    - A diferença é que a condição só é testada no final.
    - Sua sintaxe é:

```
do
   comando;
while (condição);
```

```
do
{   comando;
      ....;
   comando;
} while (condição);
```

Programando em C

- **Comandos de Repetição - Do/While**

```
#include <stdio.h>
#include <conio.h>
// Calcula bonus do Cliente
main (){
float valor ;
int cod, op;
printf("Loja Compre Tudo\n\n");
    do{
        printf("Codigo do Cliente ");
        scanf("%d", &cod);
        printf("Valor das Compras: ");
        scanf("%f", &valor);
        if (valor< 5000)
           printf("\n\nValor do bonus R$  %2.f\n", valor*0.1);
        else
           printf("\n\nValor do bonus R$  %2.f\n", valor*0.15);
        printf("\n\n Deseja continuar (1-sim/2-nao)?");
        scanf("%d",&op);
        } while (op ==1 );
}
```

Programando em C

- **Comandos Desestruturadores**
  - **Break**
    - Este comando força o encerramento de uma repetição. Sintaxe do break:

      ```
      break;
      ```
  - **Continue**
    - O comando "continue" funciona de maneira análoga ao "break", contudo ao invés de forçar o encerramento da repetição, força nova iteração saltando o código entre seu uso e a marca de término da repetição. Sintaxe do continue:

      ```
      continue;
      ```

Bons Estudos!