Computação 2

Aula 1

Estruturas básicas de programação.

Profª. Fabiany
fabianyl@utfpr.edu.br

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Plano de Aula

- Armazenamento e Recuperação de dados;
- Estruturas Estáticas;
- Tipos abstratos de dados;
- Estruturas Dinâmicas;
- Algoritmo de Pesquisa;
- Algoritmo de Ordenação.

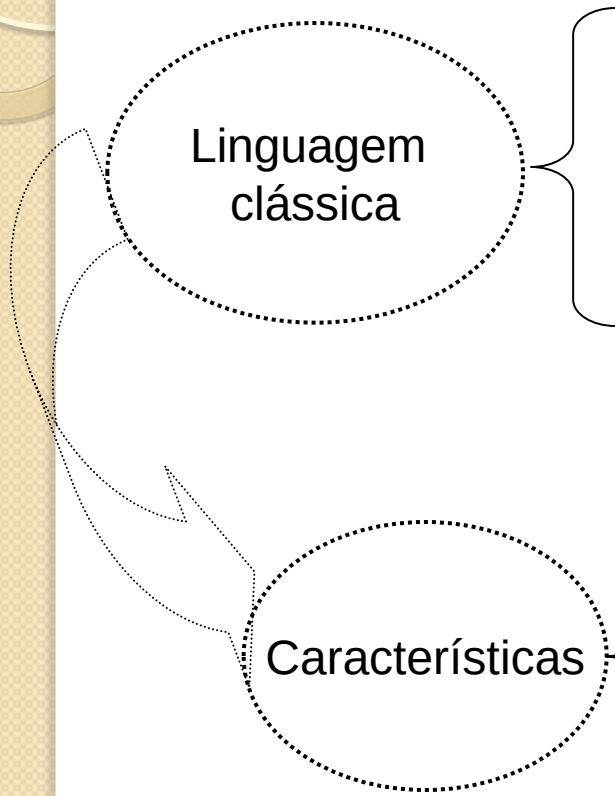
Bibliografia

- SHILDT, H. **C, Completo e Total**, 3a edição, rev. e atual. Ed. Makron. São Paulo, c1997.
- KERNIGHAM, B. W.; RITCHIE, D. M. **A Linguagem de Programação C**: padrão ANSI. Ed. Campus. Rio de Janeiro, 1989.
- FEOFILOFF, Paulo. **Algoritmos em Linguagem C**. Rio de Janeiro: Campus/Elsevier, 2009.
- TENENBAUM, Aaron M.; LANGSAM, Yedidyah; AUGENSTEIN, Moshe. **Estruturas de dados usando C**. São Paulo: Pearson Makron Books, 2005. 884 p.
- FOBERLLONE, André Luiz Villar; EBERSPÄCHER, Henri Frederico. **Lógica de programação a construção de algoritmos e estrutura de dados**. 3. ed. Makron, 2000.
- GUIMARÃES, Angelo de Moura; LAGES, Newton Alberto de Castilho. **Algoritmos e estruturas de dados.** Rio de Janeiro: LTC, 1994.
- ASCENCIO, Ana Fernanda Gomes; CAMPOS, Edilene Aparecida Veneruchi de. **Fundamentos da programação de computadores.** 2. ed. Pearson Prentice Hall, 2007.

Métodos de Avaliação

- Nota:
  - ✓ Exercícios Avaliados(30%)
  - ✓ Prova (40%)
  - ✓ Trabalho Final (30%)
  - ✓ Recuperação: prova substitutiva

Linguagem C



- **Uma linguagem difundida:**
  - Amplamente utilizada...
  - Uma linguagem veterana...
  - Sintaxe difundida, servindo como inspirações tecnológicas.

- **Uma linguagem multi-nível:**
  - Permite compor programas com abordagens variando entre ‘baixo e alto nível’
- **Organização**:
  - Funções e estruturas de informação.
- **Ponteiros:**
  - Permite a independência de memória pré-alocada.

Dennis Ritchie e Ken Thompson, anos 70

Variáveis em C

Tipo de dados	Variação	Total de Bytes Utilizados
char	0 a 255	1
int	-32.768 a 32.767	2
short int	-128 a 127	1
unsigned int	0 a 65.535	2
long int	-4.294.967.296 a 4.294.967.296	4
float	Aproximadamente 6 dígitos de precisão	4
double	Aproximadamente 10 dígitos de precisão	8
long double	Aproximadamente 10 dígitos de precisão	10
void	-	0

Variáveis e Constantes

- Variáveis e constantes são os elementos básicos que um programa manipula.
- Um programa deve conter declarações que especificam de que tipo são as variáveis que ele utilizará e as vezes um valor inicial.
- Tipos podem ser por exemplo: inteiros, reais, caracteres, etc.
- As expressões combinam variáveis e constantes para calcular novos valores.

Constantes

- Constante é um determinado valor fixo que não se modifica ao longo do tempo, durante a execução de um programa.

Variáveis

- Uma variável é um espaço reservado na memória do computador para armazenar um tipo de dado determinado, cujo conteúdo pode se alterado ao longo do tempo durante a execução de um programa.

- Variáveis devem receber nomes para poderem ser referenciadas e modificadas quando necessário.

Variáveis

- Embora uma variável possa assumir diferentes valores, ela só pode armazenar um valor a cada instante.

- As variáveis só podem armazenar valores de um mesmo tipo.

Tipos de variáveis

As variáveis e as constantes podem ser basicamente de quatro tipos:

- **Numéricas:** armazenar números, podem ainda ser classificadas como Inteiras e Reais. Inteiros apenas valores inteiros e Reais para números que possuam casas decimais.
- **Caracteres:** armazenar conjunto de caracteres. Ex. nomes.
- **Lógicas:** armazenam dados lógicos que podem ser Verdadeiro ou Falso.

Comandos em linguagem C

- Para declarar (criar) uma variável em C:

tipo nome_variavel;

- ✓ Variáveis inteiras:

  int numero;

  int valor, calculo;

- ✓ Variáveis reais:

  float conta; ou **double** conta;

  float media, valor; ou **double** media,valor;

- ✓ Variáveis literais:

  char c;

  char letra, vogal;

Variáveis em C

Tipo de dados	Variação	Total de Bytes Utilizados
char	0 a 255	1
int	-32.768 a 32.767	2
short int	-128 a 127	1
unsigned int	0 a 65.535	2
long int	-4.294.967.296 a 4.294.967.296	4
float	Aproximadamente 6 dígitos de precisão	4
double	Aproximadamente 10 dígitos de precisão	8
long double	Aproximadamente 10 dígitos de precisão	10
void	-	0

Comandos em linguagem C

- Para atribuir valor a uma variável em C utilizamos o operador **=**:

  - ✓ Variáveis inteiras:

    numero = 1;

    valor = 100;

  - ✓ Variáveis reais:

    conta = 2.8;

    media = 56.9;

  - ✓ Variáveis literais:

    letra = ‘b’;

    vogal = ‘a’;

Regras para nomes de variáveis em C

- Deve começar com uma letra (maíuscula ou minúscula) ou subscrito (*underline* _).
- Nunca pode começar com um número.
- Pode conter letras maiúsculas, minúsculas, números e subscrito.
- Não pode-se utilizar como parte do nome de uma variável

  { (+ - * / \ ; . , ?

Operadores

- Os operadores são meios pelo qual realizamos cálculos, comparações e avaliações dos dados no nosso programa. Temos três tipos de operadores:

1. Operadores Aritméticos
2. Operadores Relacionais
3. Operadores Lógicos

Operadores Aritméticos

Os operadores aritméticos são utilizados para obter resultados numéricos.

Operação	Símbolo
Adição	+
Subtração	-
Multiplicação	*
Divisão	/

Trabalhando com variáveis e operadores

- Podemos fazer uma variável receber qualquer valor numérico, assim como receber o valor de outra variável ou atualizar o próprio valor com/sem uso de operadores.
- **Exemplos:**

```
int numero;
float media, valor;

numero = 1;
numero = numero + 1;

media = 7 + 8 + 3;
media = media / 3;

valor = numero * 3;
valor = valor + 10;
```

Exemplo variáveis e operadores

- **Exemplos:**

```
float total, valor;
int quantidade, idade, n;

    n = 5;
    n = n + 1;

    idade   =  30 + 2;
    idade = 2014 – 1990;

    valor = 2.5;
    quantidade = 5;
    total = valor * quantidade;

    valor = total/quantidade;
```

Operadores Relacionais

São utilizados para comparar caracteres e números. Estes operadores sempre retornam valores lógicos (v ou f).

Descrição	Símbolo
Igual a	==
Diferente de	!=
Maior que	>
Menor que	<
Maior ou igual a	>=
Menor ou igual a	<=

Operadores Lógicos

Servem para combinar resultados das expressões.

Operador	Operação
&&	AND
||	OR
!	NOT

AND: uma expressão AND é verdadeira se todas as condições forem verdadeiras.

OR: uma expressão OR é verdadeira se pelo menos uma condição for verdadeira.

NOT: um expressão NOT inverte o valor da expressão ou condição, se verdadeira inverte para falsa e vice-versa.

Escrever na tela – printf()

- A função printf é parte de um conjunto de funções pré-definidas armazenadas em uma biblioteca padrão de rotinas da linguagem C.
- Ela permite apresentar na tela os valores de qualquer tipo de dado.
- O primeiro argumento de printf é um *string de controle*, uma seqüência de caracteres entre aspas. A lista de argumentos serão variáveis cujos valores serão formatados e apresentados na tela.

printf (string_de_controle,lista_de_argumentos);

Escrever na tela – printf()

- A string de controle mostra não apenas os caracteres que devem ser colocados na tela, mas também quais as variáveis e suas respectivas posições.
- Isto é feito usando-se especificadores de formatação, que usam a notação **%** seguido de um único caractere. Na string de controle indicamos quais, de qual tipo e em que posição estarão as variáveis a serem apresentadas.

Especificadores	Significado
%i ou %d	int (inteiro)
%c	Char (caractere)
%f	float ou double (número real)
%s	String (vetor de caractere)

Escrever na tela – printf()

Exemplos printf()

- printf (“Ola Mundo”); -> Ola Mundo
- printf ("Teste %% %%"); -> **Teste % %**
- printf ("%f",65.3); -> **65.3**
- printf ("Um caractere %c e um inteiro %d",‘A',120); -> **Um caractere A e um inteiro 120**
- printf ("%s e um exemplo","Este"); -> **Este e um exemplo**
- printf ("%s%d%%","Juros de ",10); -> **"Juros de 10%"**
- printf(“O valor de media e: %f”); -> **O valor de media e 59.6**

Ler do teclado – scanf()

- A função scanf lê dados do teclado de acordo com um formato especificado e atribui os dados recebidos a uma ou mais variáveis do programa;
- Assim como o printf(), o scanf() também usa uma string de formato pra descrever como os dados recebidos serão formatados. A string de formato utiliza os mesmos especificadores de formatação utilizados pela função printf();
- Além da string de formato, esta função recebe uma lista de argumentos, que devem ser passados por referência (precedidos pelo caractere &).

scanf (string_de_controle,lista_de_argumentos);

Ler do teclado – scanf()

- Exemplos:

- Espera que o usuário digite um inteiro. O valor digitado será o conteúdo da variável n.

  scanf("%d", &n);
- Espera que o usuário digite um valor real. O valor digitado será o conteúdo da variável valor.

  scanf("%f", &valor);
- Espera que o usuário digite dois inteiros. O primeiro valor digitado será o conteúdo da variável m e o segundo valor será o conteúdo da variável n.
- scanf("%d %d", &m, &n);

Comandos em linguagem C

- Ler valores do teclado e escrever valores na tela :
- Funções Scanf e Printf;
  - ✓ Ler variáveis inteiras:

```
printf ( " Informe um valor inteiro: \n " );
scanf ( " %i ", &numero);
```

  - ✓ Ler variáveis reais:

```
printf ( " Informe um valor real: \n " );
scanf ( " %f ", &conta);
```

  - ✓ Ler variáveis literais:

```
printf("Informe uma letra (caracter): \n");
scanf("%c", &letra);
```

Primeiro programa em C

```
#include <stdio.h>

void main()
{

    printf ( " Ola Mundo! \n " );

}
```

Comandos em linguagem C

- Para declarar uma variável em C:
  - ✓ Variáveis inteiras:

    int numero;

    int valor, calculo;
  - ✓ Variáveis reais:

    float conta; ou **double** conta;

    float media, valor; ou **double** media,valor;
  - ✓ Variáveis literais:

    char c;

    char letra, vogal;

Estrutura de Decisão

- Na maioria das vezes necessitamos tomar decisões no andamento de um programa. Essas decisões interferem diretamente no andamento do programa.

- Os comandos de decisão ou desvio fazem parte das técnicas de programação que conduzem a estruturas de programas que não são totalmente sequenciais.

- Com as instruções de salto ou desvio pode-se fazer com que o programa proceda de uma ou outra maneira, de acordo com as decisões lógicas tomadas em função dos dados ou resultados anteriores.

Se entao {} / if {}

- A estrutura de decisão “IF” normalmente vem acompanhada de um comando, ou seja, se determinada opção for satisfeita pelo comando IF então execute determinado comando.

if (condição)

comando;

- O *comando* só será executado se a *condição* for verdadeira. Uma *condição* é uma comparação que possui dois valores possíveis: verdadeiro ou falso;

Se entao {} / if {}

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

- Em C, torna-se obrigatório a utilização de chaves quando existe mais de um *comando a executar. Os comandos entre chaves { } só serão executados se* a *condição* for verdadeira.

Se entao {} / if {}

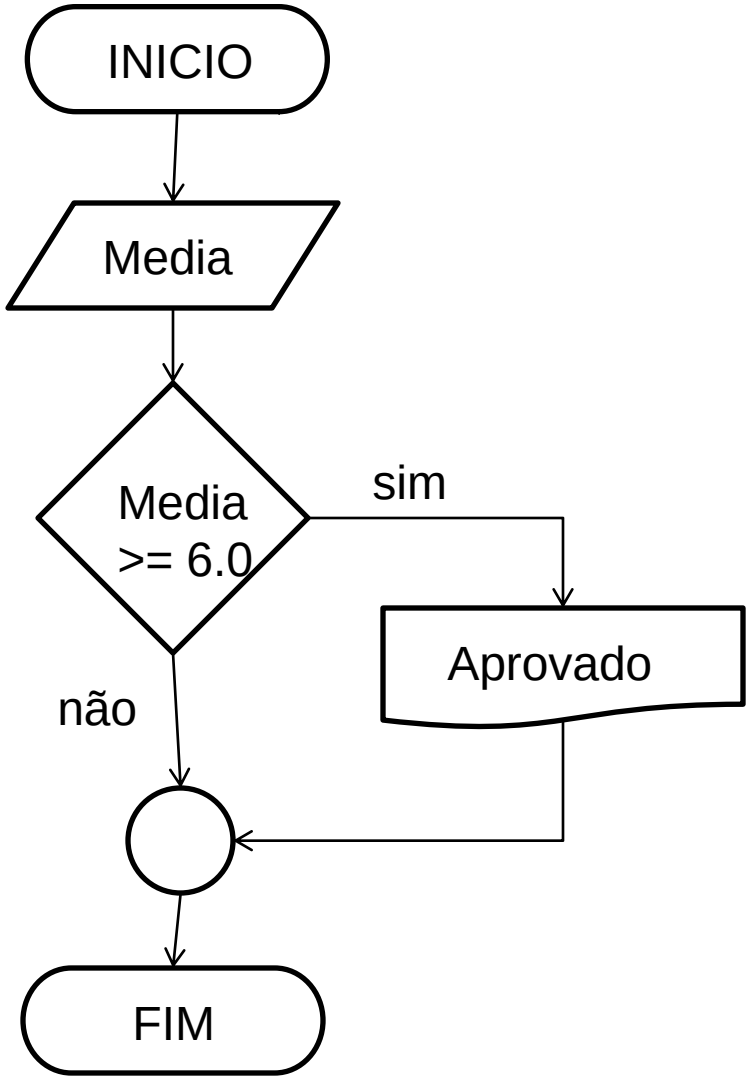
- Exemplo: Um aluno somente estará aprovado se sua média for maior ou igual a 5.0.
- Em algoritmo:

```
Se (media >= 6) então {
    escreva (“Aprovado”);
}
```

- Em C:

```
if (media >= 6) {
    printf(“Aprovado”);
}
```

Se entao {} / if {}



Operadores Relacionais

As condições do if(condição){} utiliza os operadores relacionais para comparar caracteres, números e variáveis. Estes operadores sempre retornam valores lógicos (v ou f).

Descrição	Símbolo
Igual a	==
Diferente de	!=
Maior que	>
Menor que	<
Maior ou igual a	>=
Menor ou igual a	<=

Exemplos uso dos operadores no if(){}

```
if (valor > 5)
       printf(“valor maior que 5\n”);

if (valor <= 5) {
      printf(“valor menor e igual a 5\n”);
      valor = valor + 10;
     }

if (numero == 1)
    numero = numero + 1;

if (numero != 1) {
   numero = numero + 2;
   printf(“Numero: %i”,numero);
}
```

Exemplo 1

```
#include <stdio.h>


void main()
{
    float media;

    printf("Digite a media\n");
    scanf("%f", &media);

    if ( media  >=  6 )
    {
        printf ( "Aprovado! \n" );
    }

}
```

Exemplo 2

```
#include <stdio.h>
//Programa 'Maior de dois números'

void main()
{
    int A, B;

    printf("Digite dois números\n");
    scanf("%i %i", &A, &B);

    if ( A  > B )
    {
        printf ( "A é maior que B! \n" );
    }

    if ( A < B )
    {
        printf ( "B é maior que A! \n" );
    }

   if ( A == B )
    {
        printf ( "B é igual a A! \n" );
    }
}
```

Se entao {} senao{} / if {} else {}

- Funciona exatamente como a estrutura “if”, com apenas uma diferença, com “if” somente podemos executar comandos caso a condição seja verdadeira, diferente do “if/else” pois sempre um comando será executado independente da condição, ou seja, caso a condição seja verdadeira o comando da condição será executado, caso contrário o comando da condição falsa será executado.

```
if (condição)
    comando1;
else
    comando2;
```

- Se a *condição* for verdadeira será executado o comando1; se for falsa, será executado o comando2.

Se entao {} senao{} / if {} else {}

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

- Se a *condição* for verdadeira, o comando1 e o comando2 serão executados, caso contrário, o comando3 e o comando4 serão executados.

Se entao {} senao{} / if {} else {}

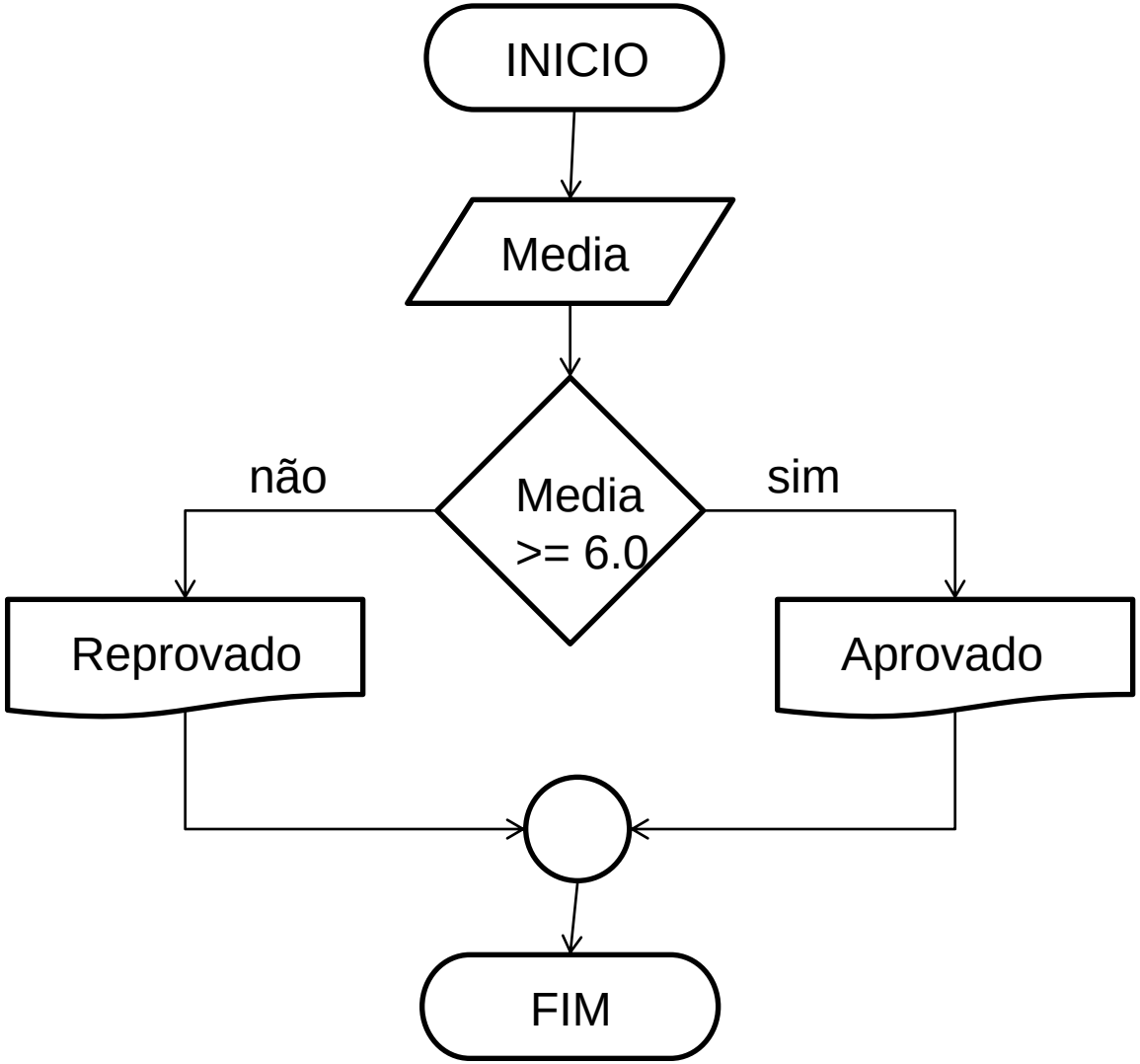
- Em algoritmo:

```
Se (media >= 6) então {
     escreva (“Aprovado”);
} senao {
    escreva (“Reprovado”);
}
```

- Em C:

```
if (media >= 6) {
    printf(“Aprovado”);
} else {
    printf(“Reprovado”);
}
```

Se entao {} senao{} / if {} else {}



Exemplo 1

```
#include <stdio.h>


void main()
{
    float media;

    printf("Digite a media\n");
    scanf("%f", &media);

    if ( media  >=  6 )
    {
        printf ( "Aprovado! \n" );
     } else {
        printf ( "Reprovado! \n" );
    }

}
```

Exemplo 2

```
#include <stdio.h>

void main()
{
    int A, B;

    printf("Digite dois números\n");
    scanf("%i %i", &A, &B);

    if ( A  > B )  {

        printf ( "A é maior que B! \n" );

     } else {

          if ( A < B )   {

           printf ( "B é maior que A! \n" );

           } else {

              // if ( A == B )
                 printf ( "B é igual a A! \n" );
           }
    }

}
```

Operadores Lógicos

Servem para combinar resultados das expressões.

Operador	Operação
&&	AND
||	OR
!	NOT

AND: uma expressão AND é verdadeira se todas as condições forem verdadeiras.

OR: uma expressão OR é verdadeira se pelo menos uma condição for verdadeira.

NOT: um expressão NOT inverte o valor da expressão ou condição, se verdadeira inverte para falsa e vice-versa.

Operadores Lógicos

- Os operados lógicos && (e), || (ou) e ! (não) são usados para conjunção, disjunção e negação, respectivamente.

Tabela && (e)	Tabela || (ou)	Tabela ! (não)
V e V = V	V e V = V	Não V = F
V e F = F	V e F = V	Não F = V
F e V = F	F e V = V	
F e F = F	F e F = F	

Exemplos uso dos operadores lógicos

```
if (valor > 5)
    printf("valor maior que 5\n");
```

- **No exemplo acima, existe apenas uma condição que, obrigatoriamente, deve estar entre parênteses.**

```
if (x > 5 && x < 10 ) {
    printf("Números entre 5 e 10");
    valor = valor + 10;
   }
```

- **No exemplo acima, existe mais de uma condição, as quais, obrigatoriamente, deve estar entre parênteses.**

```
if (x == 5 && (y == 2 || y == 3))
   printf("x é igual a 5, e y é igual a 2 ou y é igual a 3");
```

- **No exemplo acima, existe mais de uma condição e mais de um operador lógico, logo, além dos parênteses que envolve todas as condições, devem existir ainda parênteses, que indiquem a prioridade de execução de condições.**

escolha caso{} / switch case{}

- A estrutura de decisão escolha/caso é utilizada para testar na condição, uma única expressão, que produz um resultado, ou, então, o valor de uma variável, em que está armazenado um determinado conteúdo. Compara-se, então, o resultado obtido no teste com os valores fornecidos em cada cláusula “caso”.

escolha caso{} / switch case{}

```
switch (variável)
{
  case valor1 : { lista de comandos; } break;
  case valor2 : { lista de comandos; } break;
   ....
  default: {lista de comandos; }
}
```

- O comando *switch (variável)* analisa o valor de uma variável para decidir qual *case* será executado. Cada *case* está associado a UM possível valor da variável , que deve ser obrigatoriamente do tipo **int** ou **char**. O comando *break* deve ser utilizado para impedir a execução dos comandos definidos nos cases subsequentes.

escolha caso{} / switch case{}

- Em algoritmo:

```
Inicio{
    inteiro valor;
    escreva("Digite um valor");
    leia (valor);
    escolha (valor) {
        caso 1: { printf("Voce digitou 1"); }
        caso 2: { printf("Voce digitou 2"); }
        outrocaso : printf("Valor zero ou maior que 2");
   }fim.
```

Exemplo de um programa

```
#include <stdio.h>
//Programa 'Sexo da Pessoa'
void main()
{
     int A;

    printf ( "Informe seu sexo : \n" );
    printf ( "1 p/ Masculino, 2 p/ Feminino \n" );
    scanf ( "%i", &A );

    switch ( A )
    {
        case 1: {
                  printf ( "Masculino! \n" );
                  }   break;

        case 2: {
                  printf ( "Feminino! \n" );
                  } break;

        default: {
                  printf ( "Erro! \n" );
                  }
    }

}
```

Exercícios

1) Faça um programa que leia dois números e mostre o maior.

2) Faça um algoritmo que receba o código e o salário de um funcionário. Conforme o código mostrar qual é o cargo, o valor do aumento (salário * percentual) e o novo salário (salário + aumento) . Os cargos estão na tabela abaixo:

Código	Cargo	Percentual
1	Escriturário	50%
2	Secretário	35%
3	Caixa	20%
4	Gerente	10%
5	Diretor	Não tem aumento

Exercícios

3) Faça um programa que leia 3 notas, calcule e mostre a média e o conceito conforme a tabela abaixo:

Média	Conceito
>=80 e <=100	A
>=70 e < 80	B
>= 60 e < 70	C
>= 50 e < 60	D
>= 0 e < 50	E

Estrutura de Repetição

- Utilizamos os comandos de repetição quando desejamos que um determinado conjunto de instruções ou comandos sejam executados um número definido ou indefinido de vezes, ou enquanto um determinado estado de coisas prevalecer ou até que seja alcançado.

enquanto faça{} / while(){}

- Para número indefinido de repetições e teste no inicio.
- É utilizada quando não se sabe o número de vezes que um trecho do algoritmo vai ser repetido, embora também possa ser utilizada quando se conhece esse número.
- Essa estrutura baseia-se na análise de uma condição. A repetição será feita enquanto a condição for verdadeira.

Linguagem C : while(){}

- Sintaxe da estrutura:

```
while (condição) {
        comando1;
        comando2;
        comando3;
        ...
}
```

- Enquanto a *condição* for verdadeira, os *comandos* que estão dentro das chaves serão executados (comando1, comando2, comando3...).

enquanto faça{} / while(){}

- Em algoritmo:

```
Inicio {
    inteiro x,y;
    x = 0;
    y = 10;
     enquanto (x < y) faca {
          escreva (x);
           x = x + 2;
       }
  } fim.
```

- Em C:

```
void main() {
        int x,y;
        x = 0;
        y = 10;
        while (x < y) {
              printf("x: %i",x);
               x = x + 2;
     }
   }
```

Linguagem C : while(){}

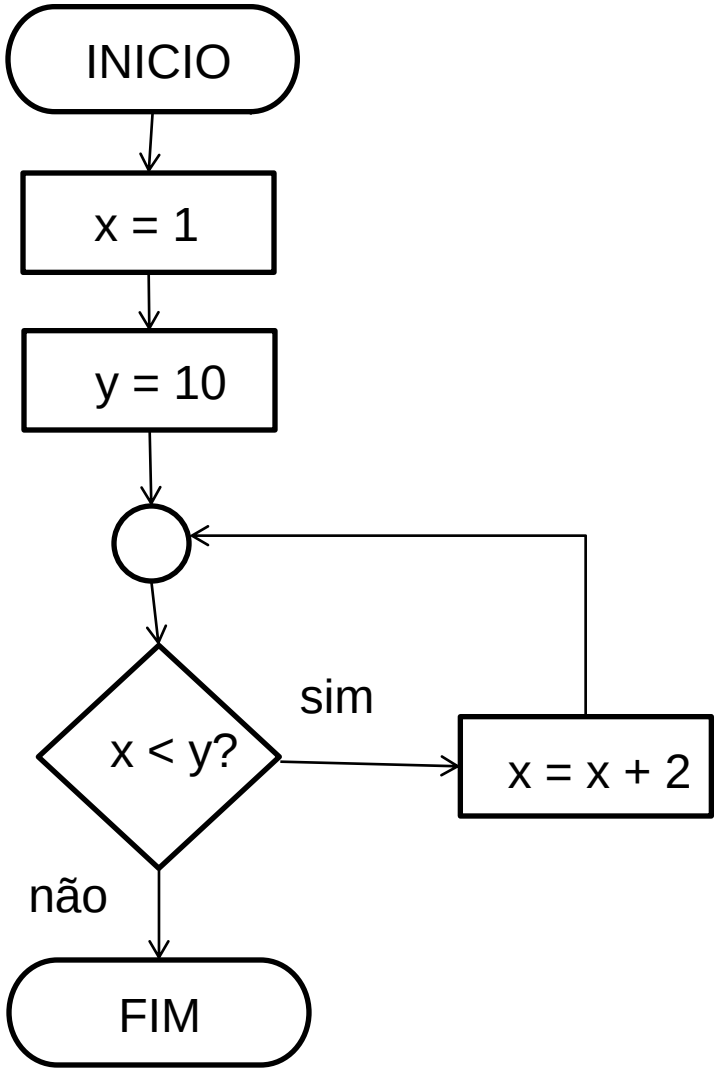
- Nos programas apresentados no slide anterior, os comandos *printf("x:%i",x);* e *x=x+2;* serão executados cinco vezes. O teste condicional avaliará *x* valendo *0, 2, 4, 6, 8, e 10*.

- Resultado:

Impressão na tela	***Variável x***
x: 0	***0***
x: 2	***2***
x: 4	***4***
x: 6	***6***
x: 8	***8***
	10

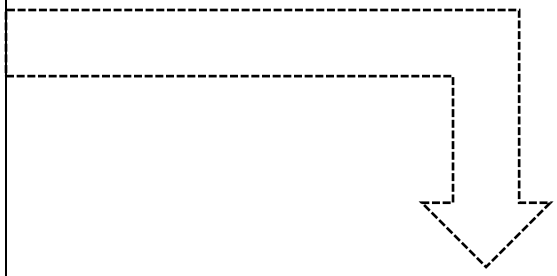
Valor de x depois de sair da estrutura = 10

Estrutura while(){}



Exemplo de um programa

```
#include <stdio.h>

//Programa 'Imprime 10 números'
void main()
{

    int A;

    A = 1;

    while ( A <= 10 )
    {
      printf ( "Número :%i \n ", A );

      A = A + 1;
    }


}
```



Impressão na tela	Valor da variável A
Número : 1	1
Número : 2	2
Número : 3	3
Número : 4	4
Número : 5	5
Número : 6	6
Número : 7	7
Número : 8	8
Número : 9	9
Número : 10	10
	11

Exercícios – while()

1) Faça um programa para imprimir 100 números na tela.

2) Faça um programa para imprimir só os números pares entre 1 e 1000(inclusive).

3) Faça um programa que leia qualquer número e diga se é par ou impar. O programa deve ficar repetindo isso até o usuário digitar o valor 0 para sair.

repita {} ate(); / do {} while();

- Para número indefinido de repetições e teste no final.
- A diferença entre a estrutura enquanto e a estrutura repita é que nesta ultima os comandos serão repetidos pelo menos uma vez, já que a condição de parada se encontra no final.

Linguagem C : do{} while();

- Sintaxe da estrutura:

```
do {
      comando1;
      comando2;
      comando3;
      ...
} while (condição) ;
```

- Os *comandos* que estão dentro das chaves serão repetidos (comando1, comando2, comando3...) até a *condição* assumir valor falso.

repita {} ate(); / do {} while();

- Em algoritmo:

```
Inicio {
    inteiro x,y;
    x = 1;
    y = 5;
    repita {
        x = x + 2;
        y = y + 1;
        escreva(x,y);
    } ate (x >= y);
} fim.
```

- Em C:

```
void main() {
        int x,y;
        x = 1;
        y = 5;
        do {
            x = x + 2;
            y = y + 1;
            printf("%i - %i\n",x,y);
        } while ( x <= y) ;
    }
```

Linguagem C : do{}while();

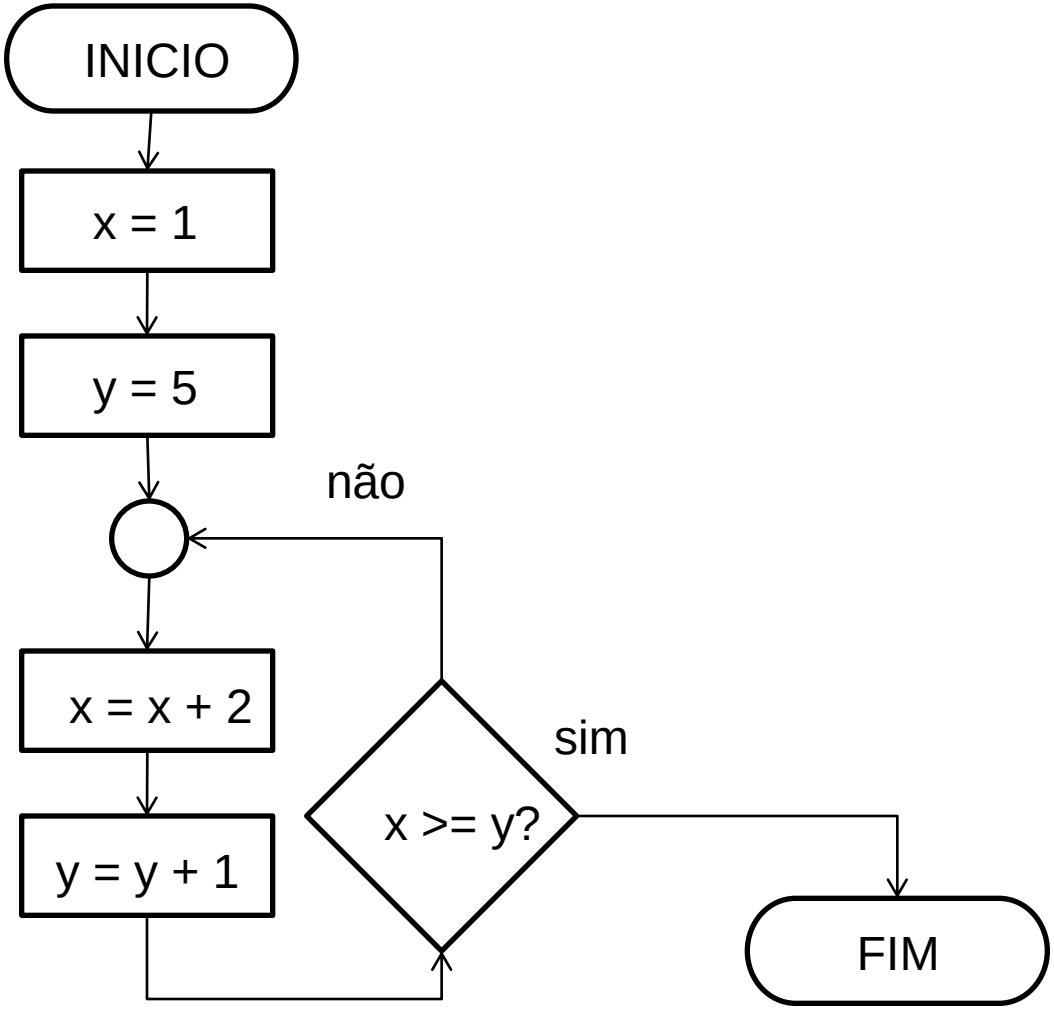
- Nos programas apresentados no slide anterior, os comandos printf("%i -%i\n",x,y);, *x = x + 2;* e y = y + 1; serão executados cinco vezes. O teste condicional avaliará *x* valendo *3, 5, 7, 9, e 11* e *y valendo 6,7,8,9,10*.

- Resultado:

Impressão na tela	***Variável x***	***Variável y***
	1	***5***
3 - 6	***3***	***6***
5 - 7	***5***	***7***
7 - 8	***7***	***8***
9 - 9	***9***	***9***
	11	***10***

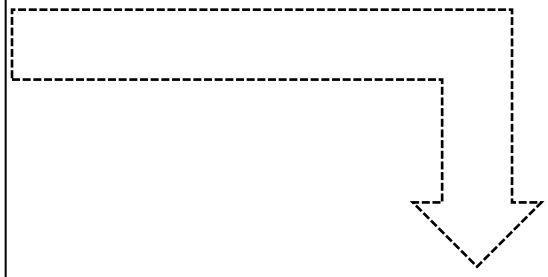
Valor de x depois de sair da estrutura = 11 e de y = 10

repita {} ate(); / do {} while();



Exemplo de um programa

```
#include <stdio.h>

//Programa 'Imprimir 10 números'
void main()
{

    int A;

    A = 1;

    do
    {
      printf ( " Número :%i\n ", A );

      A = A + 1;

    }  while ( A <= 10 );

}
```



Impressão na tela

```
Número : 1
Número : 2
Número : 3
Número : 4
Número : 5
Número : 6
Número : 7
Número : 8
Número : 9
Número : 10
```

Valor da variável A

```
1
2
3
4
5
6
7
8
9
10
11
```

Exercícios – do{} while();

1) Faça um programa para imprimir 200 números na tela.

2) Faça um programa para imprimir só os números impares entre 1 e 1000(inclusive).

3) Faça um programa que leia qualquer número e diga se é par ou impar. O programa deve ficar repetindo isso até o usuário digitar o valor 0 para sair.

para ate faça{ } / for () { }

- Para número definido de repetições.
- Essa estrutura de repetição é utilizada quando se sabe o numero de vezes que um trecho do algoritmo deve ser repetido.
- O incremento, ou seja, o contador é adicionado automático.

Linguagem C : for(;;){ }

- Sintaxe da estrutura:

for (*i = valor inicial*; *condição*; *incremento ou decremento de i*) {

comando1;

comando2;

...

}

- A primeira parte (*i=valor inicial*) atribui um valor inicial à variável *i* (pode ser qualquer variável), que tem como função controlar o número necessário de repetições.
- A segunda parte (*condição*) corresponde a uma expressão relacional, que quando assumir o valor falso, determinará o fim da repetição.
- A terceira parte (*incremento ou decremento de i*) é responsável por alterar o valor da variável *i* com o objetivo de , em algum momento, fazer com que a condição assuma o valor falso.

para ate faça{ } / for () { }

- Em algoritmo:

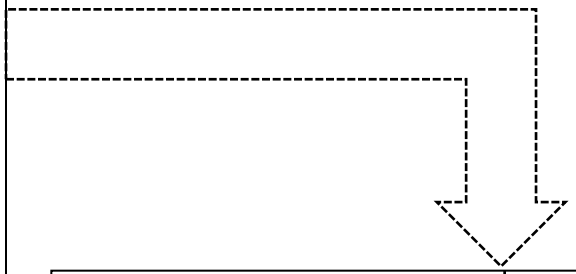
```
Inicio {
  inteiro num, soma;
  para num = 1 ate 100 faca {
        escreva (num);
        soma = soma + num;
   }
  escreva(soma);
} fim.
```

- Em C:

```
void main() {
      int n, soma;
      for (n = 1; n <= 100; n = n+1) {
                printf("n: %i",n);
                 soma = soma + n;
        }
      printf("%i",soma);
    }
```

Exemplo de um programa

```
#include <stdio.h>

//Programa 'Imprime 10 números'

void main()
{
    int A;


    for ( A = 1; A <= 10; A = A + 1 )
    {
      printf ( " Número %i : \n ", A );
    }


}
```



Impressão na tela

Número : 1
Número : 2
Número : 3
Número : 4
Número : 5
Número : 6
Número : 7
Número : 8
Número : 9
Número : 10

Valor da variável A

1
2
3
4
5
6
7
8
9
10
11

Exercícios

1) Faça um programa que leia um valor m. Calcular a soma de todos os números entre 1 até o valor m.
2) Faça um programa que leia o valor n e calcule o fatorial desse número.
3) Faça um programa para somar os números pares entre 5 e 500 (inclusive).
4) Faça um programa que calcula a associação em paralelo de dois resistores R1 e R2 entrados pelo usuário via teclado. O programa fica pedindo estes valores e calculando até que o usuário entre com um valor de resistência igual a zero. Fórmula: R=R1*R2/(R1+R2)

Exercícios

- Faça um programa que receba o código e o salário de vários funcionários, até que o usuário digite 0 para o código para sair. Conforme o código mostrar qual é o cargo, o valor do aumento (salário * percentual) e o novo salário (salário + aumento) de cada funcionário . Os cargos estão na tabela abaixo:

Código	Cargo	Percentual
1	Escriturário	50%
2	Secretário	35%
3	Caixa	20%
4	Gerente	10%
5	Diretor	Não tem aumento

Exercícios

- Faça um programa que leia 10 notas, calcule e mostre a média e o conceito conforme a tabela abaixo:

Média	Conceito
>=80 e <=100	A
>=70 e < 80	B
>= 60 e < 70	C
>= 50 e < 60	D
>= 0 e < 50	E

Referências Bibliográficas

- Baseado nos slides do Professor Jean Simão disponível em: http://www.pessoal.utfpr.edu.br/jeansimao/index.htm

- Ascencio, A. F. G., CAMPOS, E. A. V. Fundamentos da programação de computadores. 2. ed. Pearson Prentice Hall, 2007.

- Forbellone A. L. V., Eberspächer, H. F.: Lógica de Programação : A construção de Algoritmos e Estruturas de Dados. Makron Books, 1993.