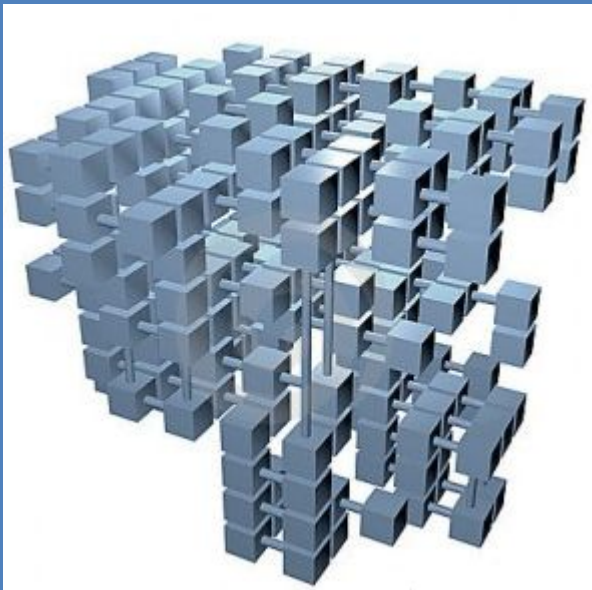


Engenharia da Computação



Algoritmo e Estrutura de Dados I Parte 7

Professor Sandro Teixeira Pinto
E-mail: sandropinto21@gmail.com

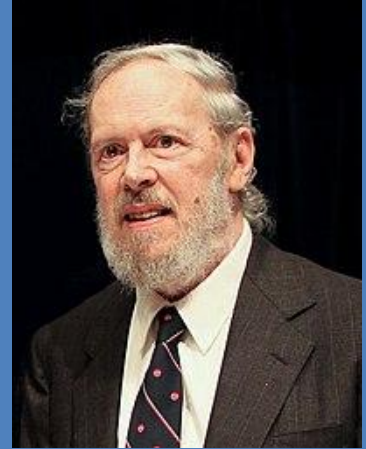
Índice

- Linguagem C++;
 - Características
- Construção de um programa na linguagem;
- Tipos de compiladores;
- Declaração de variáveis;
- Comandos de Entrada;
- Comandos de Saída;
- Operações e funções;
- Estruturas;

C/C++

- **Objetivo Geral:**
 - Apresentar a linguagem de programação C/C++.

A Linguagem C.



- Dennis Ritchie
 - 1972 (Centro de Pesquisas da *Bell Laboratories*)
- Primeira Utilização
 - Sistema Operacional UNIX.
- Linguagem de propósito geral, sendo adequada à programação estruturada.
 - Compiladores, analisadores léxicos(Scanner, analisados de caracter), bancos de dados, editores de texto, etc.

A Linguagem C.

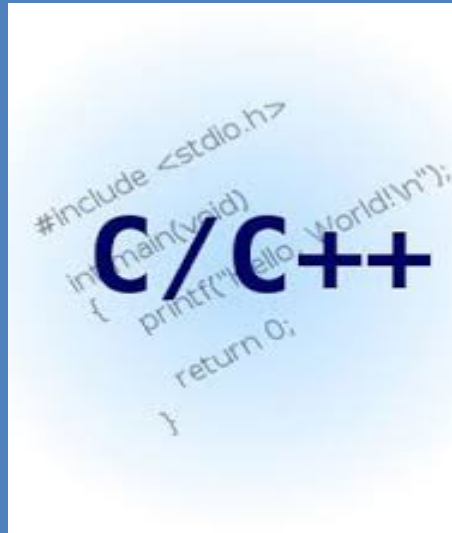
- Características da Linguagem
 - Portabilidade
 - Modularidade
 - Compilação separada
 - Recursos de baixo nível
 - Geração de código eficiente
 - Confiabilidade
 - Regularidade
 - Simplicidade
 - Facilidade de uso.

A Linguagem C++.

- C é uma das linguagens de programação mais populares;
- Existem poucas arquiteturas para as quais não existem compiladores para C.
- C tem influenciado muitas outras linguagens de programação, mais notavelmente C++, que originalmente começou como uma extensão para C.

A Linguagem C++.

- A linguagem de programação C++ foi originalmente derivada do C para suportar **programação orientada a objetos**.



Construção de um programa em C++

- Escrito em linguagem de alto nível, utilizando:
 - Letras, números e outros símbolos (\$,%,#...)
- Para que uma máquina entenda o programa-fonte, é necessário um processo de tradução do código-fonte para o código de máquina ou binário.
 - 0100 1001 0011 0110 1101...
 - Operação realizada automaticamente pelo compilador.

Construção de um programa em C++

- Execução de um programa em C/C++;
 - Utilizar um **editor de texto** para escrever o **código-fonte** e guardá-lo em um arquivo chamado **arquivo-fonte** ou **código-fonte**. (Extensão .cpp)
 - Compilar o código-fonte. (Traduz o código-fonte para linguagem de máquina).
 - Arquivo programa ou código executável. (Extensão .exe para Windows)

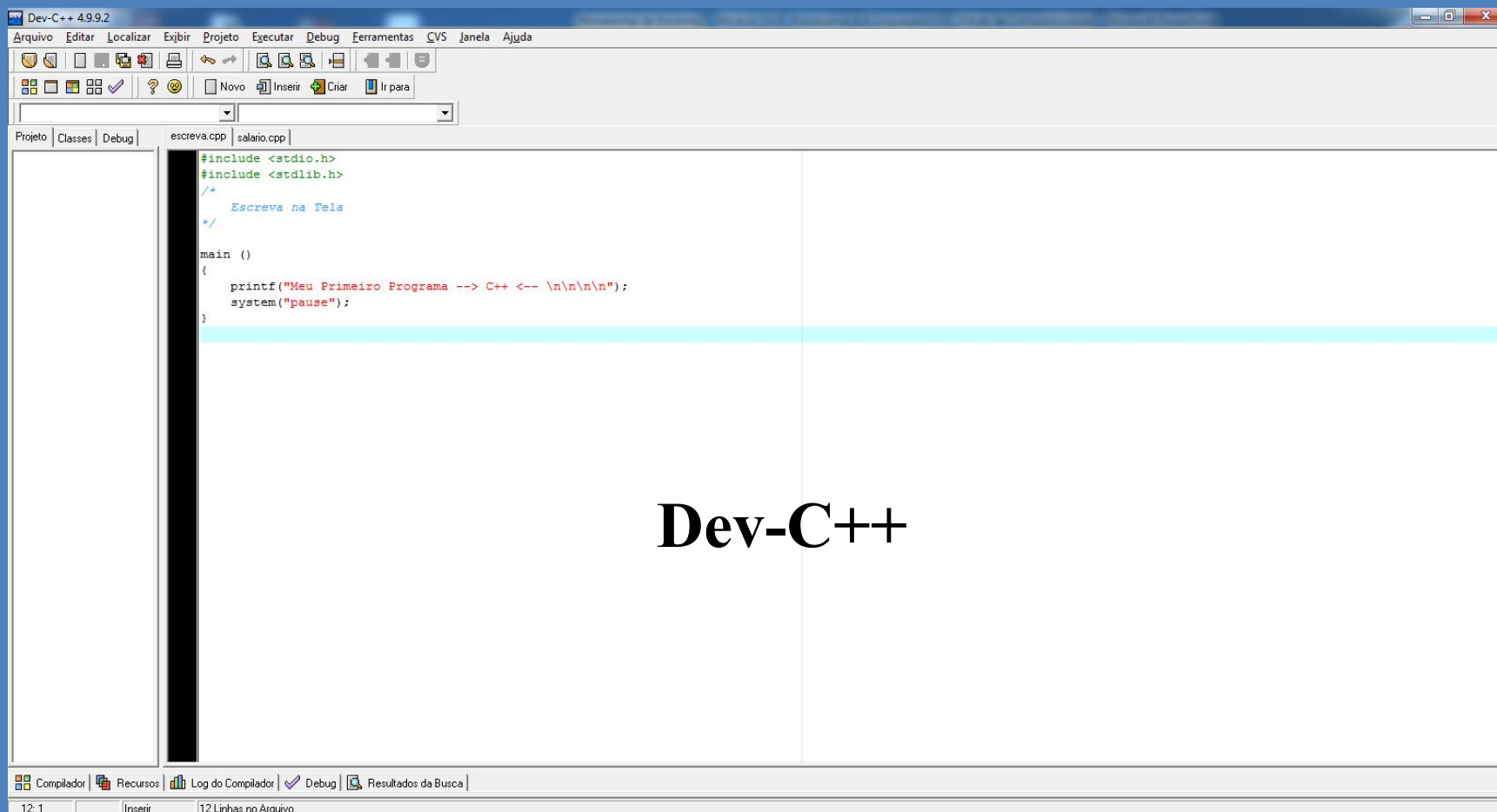
Construção de um programa em C++

- Analise do Problema
- Algoritmo (Fluxograma / Pseudocódigo)
- Código-fonte C++
(nome.cpp)
- Compilador C++
- Programa Executável
(nome.exe)

Tipos de Compiladores

- Compilador de linha de ordens ou compilador ativo.
- Compilador incluído em um ambiente Integrado de Desenvolvimento (IDE – Integrated Development Environment).
 - Usamos o Dev-C++.
 - Dev-C++ é um Ambiente de Desenvolvimento Integrado (IDE - Integrated Development Environment) para programação na linguagem C/C++.

Tipos de Compiladores




Dev-C++

Edição de um Programa


- Editores de texto
 - Microsoft Windows
 - Bloco de notas;
 - Microsoft Office Word;
 - Notepad;
 - ...
 - Linux
 - Vi;
 - Vim;
 - Nano;
 - OpenOffice Writer ;
 - ...
 - Devemos ter a preocupação de guardar o arquivo em formata (ASCII)
 - SALVAR COMO □ “nome.ccp” para compilação no Dev-c++

Estrutura Sequencial em C/C++

```
# include <nome_da_biblioteca>
int main()
{
    bloco_de_comandos;
    return 0;
}
```



```
import java.util.ArrayList;
import java.util.List;
```



```
using System.Linq;
using System.Text;
using System.Web;
using System.Web.UI;
```

* Bibliotecas são arquivos contendo várias funções que podem ser incorporadas aos programas escritos em C/C++.

Declaração de variáveis em C/C++

As variáveis são declaradas após a especificação de seus tipos.

Para números inteiros □ int;

Para números reais □ float;

Para um caractere □ char;

Exemplos

float x;

Int y, x;

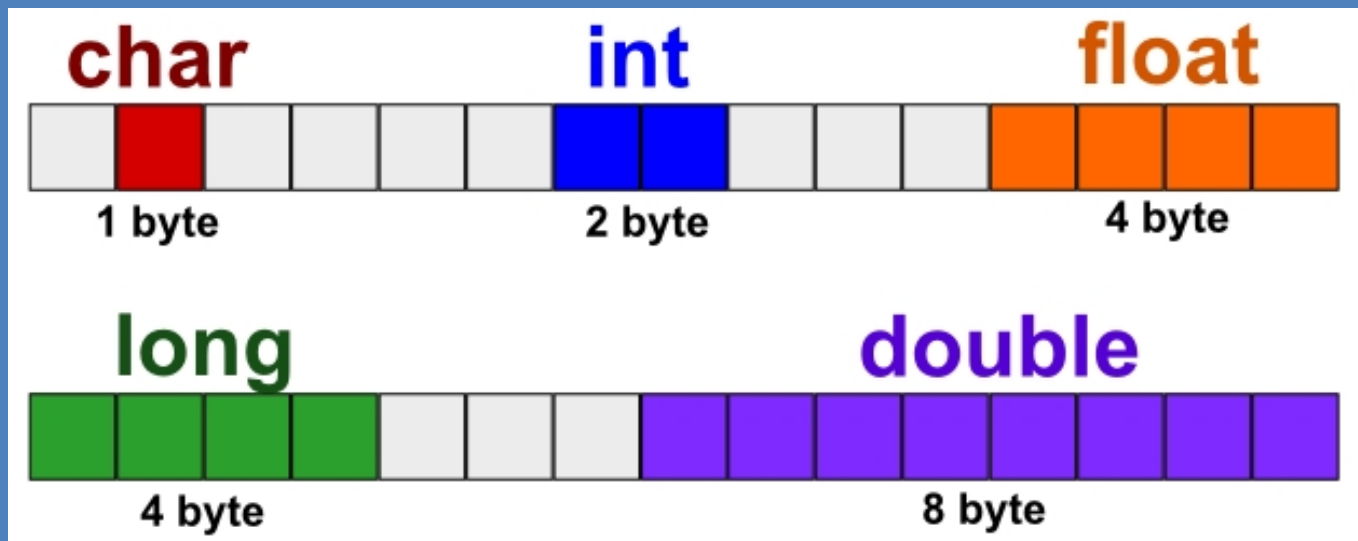
char sexo; { 0 ou 1 }

char nome[40]; {Pode armazenar até 39 caracteres}

Declaração de variáveis em C/C++

Tipo	Tamanho	Intervalo
Char	8 bits	-127 a 128
Unsigned char	8 bits	0 a 255
signed char	8 bits	-127 a 128
short int	16 bits	-32768 a 32767
unsigned short int	16 bits	0 a 65535
signed short int	16 bits	-32768 a 32767
int	32 bits	-2.147.483.648 a 2.147.483.647
signed int	32 bits	-2.147.483.648 a 2.147.483.647
unsigned int	32 bits	0 a 4.294.967.295
long int	32 bits	-2.147.483.648 a 2.147.483.647
signed long int	32 bits	-2.147.483.648 a 2.147.483.647
unsigned long int	32 bits	0 a 4.294.967.295
float	32 bits	$3,4 \times 10^{-38}$ a $3,4 \times 10^{+38}$
Double	64 bits	$1,7 \times 10^{-308}$ a $1,7 \times 10^{+308}$
Long Double	80 bits	$3,4 \times 10^{4932}$ a $1,1 \times 10^{4932}$

Declaração de variáveis em C/C++



Declaração de constantes em C/C++

As constantes são declaradas depois das bibliotecas e seus valores não podem ser alterados durante a execução do programa.

```
#define nome valor
```

```
#define x 7
```

```
#define y 4.8
```

```
#define nome "Maria"
```

Declaração de atribuições em C/C++

É utilizado para conceder valores ou operações a variáveis;

```
x = 4;
```

```
x = x + 2;
```

```
y = 2.5;
```

```
sexo = 'F';
```

- Caracteres são representados entre apóstrofes ('), As cadeias de caracteres devem ser entre aspas (“)
- Cada comando é finalizado com (;)

Comandos de entrada em C/C++

Utilizados para receber dados digitados pelo usuário, os dados são armazenados em uma variável;

`scanf(“%d “,&x);`{Um número *inteiro* será armazenado na variável *x*}

`scanf(“%f “,&z);`{Um número *real* será armazenado na variável *z*}

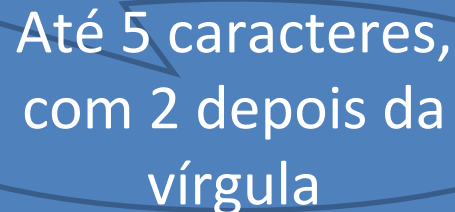
`scanf(“%s “,&nome);`{*Um ou mais caracteres* serão armazenados na variável *nome*}

`scanf(“%c “,&sexo);`{Um *caractere* é armazenados na variável *sexo*}

Comandos de saída em C/C++

Utilizado para mostrar os dados na tela ou impressora;

- **printf**("%d", y); {Mostra um número inteiro armazenado na variável y}
- **printf**("Conteúdo de y = %d", y);
- **printf**("%f", x); {*Mostra um número real armazenado na variável x*}
- **printf**("%5.2f", x); {*Mostra um número **real** armazenado na variável x utilizando 5 caracteres da tela exemplo (51215.25) 25.80, 29.14*}



Até 5 caracteres,
com 2 depois da
vírgula

Comentários em C/C++

Comentários são textos que podem ser inseridos em programas com o objetivo de documentá-los, eles não são analisados pelo compilador.

/*

Linhas de comentários...

Linhas de comentários...

*/

```
/*  
Classe "Comentários2"  
  
Todo este texto esta sendo ignorado pelo Java e tem como única função explicar o  
funcionamento do código do programa.  
*/
```

Operações e funções predefinidas em C/C++

Operador	Exemplo	Comentário
=	x = y	O conteúdo da variável y é atribuído à variável x.
+	x + y	Soma o conteúdo de x e de y
-	x - y	Subtrai o conteúdo de y do conteúdo de x
*	x * y	Multiplica o conteúdo de x pelo conteúdo de y
/	x / y	Obtém o quociente da divisão de x por y int z = 5 / 2; ? A variável z receberá o valor 2 float z = 5.0 / 2.0; ? A variável z receberá o valor 2.5
%	x % y	Obtém o resto da divisão de x por y O operador % só pode ser usado com operandos do tipo inteiros

Operações e funções predefinidas em C/C++

Operador	Exemplo	Comentário
<code>+=</code>	<code>x += y</code>	Equivale <code>x = x + y</code> .
<code>-=</code>	<code>x -= y</code>	Equivale <code>x = x - y</code> .
<code>*=</code>	<code>x *= y</code>	Equivale <code>x = x * y</code> .
<code>/=</code>	<code>x /= y</code>	Equivale <code>x = x / y</code> .
<code>%=</code>	<code>x %= y</code>	Equivale <code>x = x % y</code> .
<code>++</code>	<code>x++</code>	Equivale <code>x = x + 1</code> .
<code>--</code>	<code>x--</code>	Equivale <code>x = x - 1</code> .

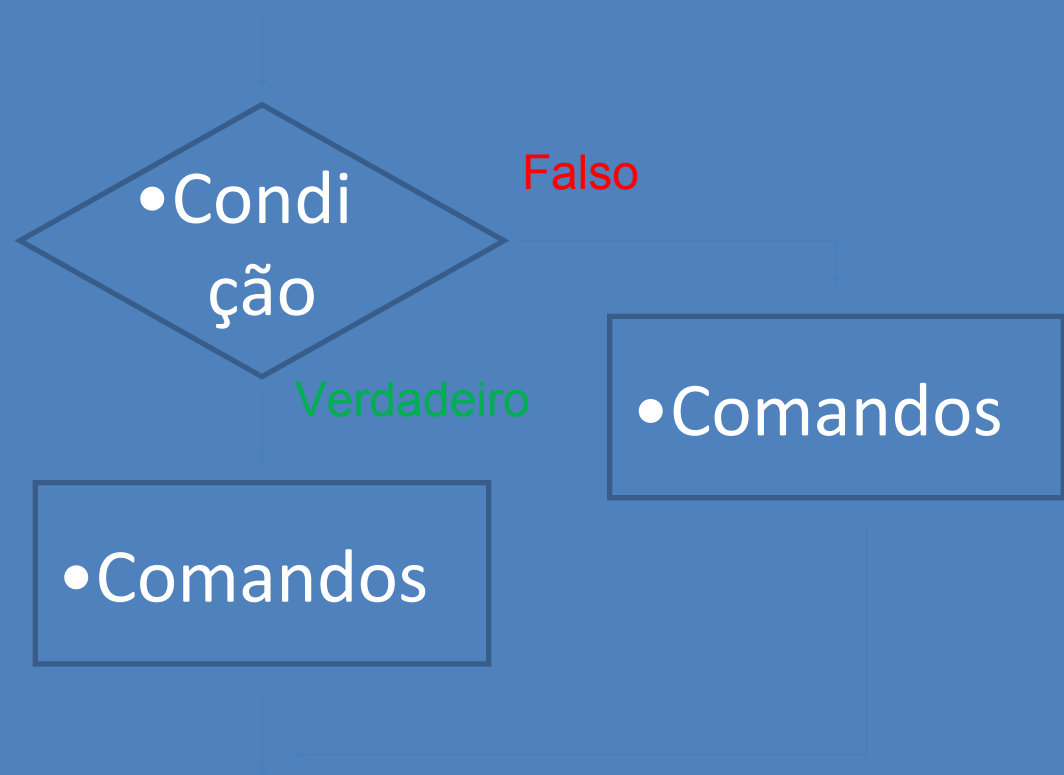
Operações e funções predefinidas em C/C++

Operador	Exemplo	Comentário
==	x == y	O conteúdo de x é igual ao conteúdo de y
!=	x != y	O conteúdo de x é diferente do conteúdo de y
<=	x <= y	O conteúdo de x é menor ou igual ao conteúdo de y
>=	x >= y	O conteúdo de x é maior ou igual ao conteúdo de y
<	x < y	O conteúdo de x é menor do que o conteúdo de y
>	x > y	O conteúdo de x é maior do que o conteúdo de y

Estrutura Condicional

- A estrutura condicional de um algoritmo pode ser **simples** ou **composta**.

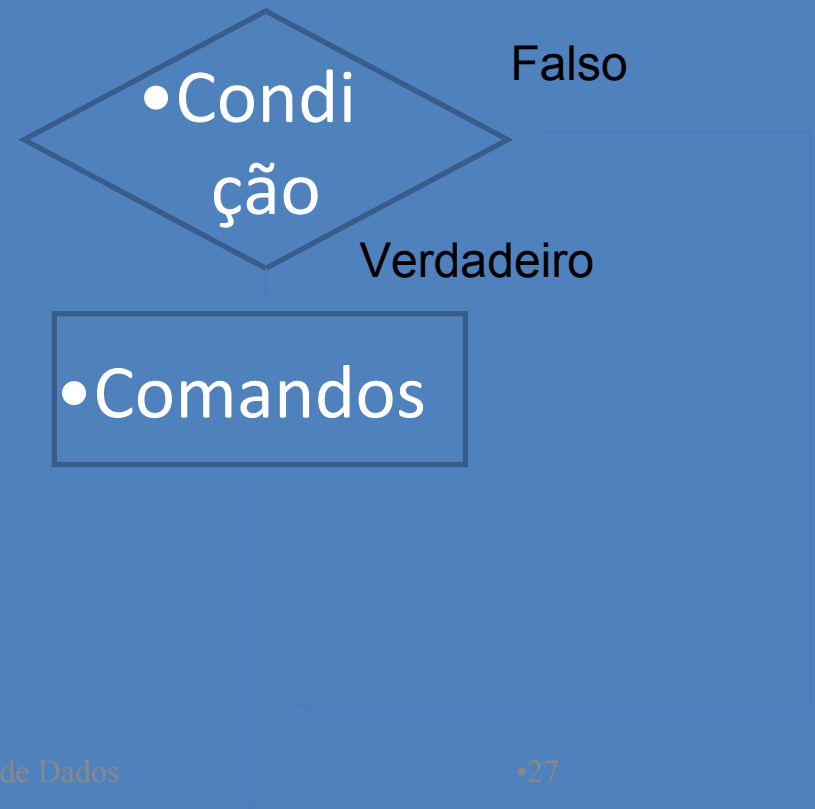
- Essa função requer a seguinte estrutura:
SE (**Teste_lógico**,
Valor_se_verdadeiro,
Valor_se_falso)



Estrutura Condicional Simples

- O comando só será executado se a condição for verdadeira.

```
If (condição) {  
    comandos  
}
```



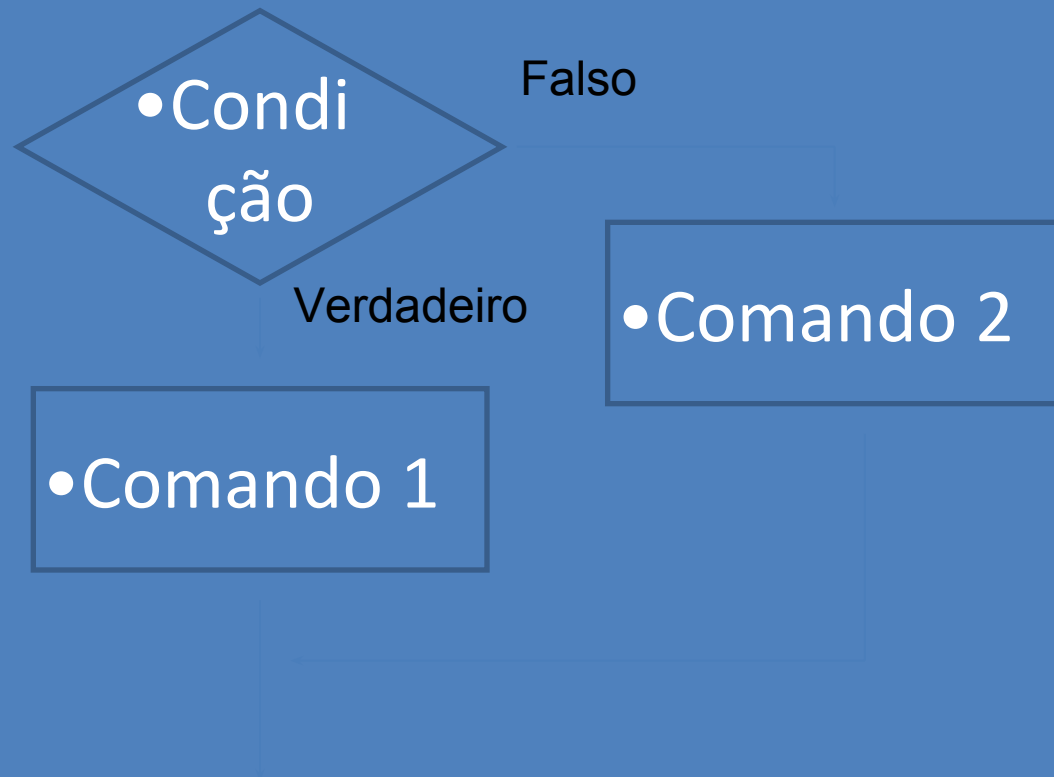
Estrutura Condicional Simples

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  main()
4  {
5      int numero;
6      printf("Digite um número : ");
7      scanf("%d", &numero);
8      if ( numero > 10 ) {
9          printf("O número %d é maior que 10 \n\n\n", numero);
10     }
11     system("pause");
12 }
```

Estrutura Condicional Composta

- Se condição for verdadeira, será executado o comando1; caso contrário, será executado o comando2.

```
If (condição) {  
    comandos 1  
} else {  
    comandos 2  
}
```



```
main ()  
{  
    int num1, num2;  
  
    cout << "Numero 1: ";  
    cin >> num1;  
  
    cout << "Numero 2: ";  
    cin >> num2;  
  
    if (num1 > num2){  
        cout << num1 << " Maior do que " << num2;  
    }else  
        cout << num2 << " Maior do que " << num1;  
}
```

Estrutura Condicional Composta

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  main()
4  {
5      int numero;
6      printf("Digite um número : ");
7      scanf("%d",&numero);
8      if ( numero > 10 )
9      {
10         printf("O número %d é maior que 10 \n\n\n",numero);
11     }else {
12         printf("O número %d é menor que 10 \n\n\n",numero);
13     }
14     system("pause");
15 }
```

Estrutura Condicional Composta

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  main()
4  {
5      int numero;
6      printf("Digite um número : ");
7      scanf("%d",&numero);
8      if ( numero > 10 ) {
9          printf("O número %d é maior que 10 \n\n\n",numero);
10         } else if(numero < 10){
11             printf("O número %d é menor que 10 \n\n\n",numero);
12         } else {
13             printf("O número %d é igual que 10 \n\n\n",numero);
14         }
15         system("pause");
16     }
```

Estrutura de repetição - FOR

- As estruturas de repetições são muito importantes para programas, pois muitas vezes o mesmo procedimento têm que ser executados mais de uma vez.
- Em C/C++, basicamente existem três tipos de estrutura de repetição: FOR(PARA), WHILE (ENQUANTO) e DO WHILE. Nessa seção estudaremos o FOR.

Estrutura de repetição - FOR

- O FOR como qualquer iteração* precisa de uma variável para controlar os loops (voltas). Em for, essa variável deverá ser inicializada, indicada seu critério de execução, e forma de incremento ou decremento. Ou seja, for precisa de três condições. Vale salientar que essas condições são separadas por ponto-e-vírgula (;).
- *ITERAÇÃO é sinônimo de repetição.

Estrutura de repetição - FOR

- **FOR (inicialização; até quando irá ser executado; incremento ou decremento).**
- No exemplo, faremos um programa que contará de 1 a 100. Você verá que é um programa que faz algo muito simples, ele soma um a uma variável várias vezes.
- O x começa em 1, o loop só será executado enquanto x for menor ou igual a 100 e a cada loop será somado 1 a x. Veja:

Estrutura de repetição – FOR

Exemplo 1

- **Setlocale** serve para adaptar o código a língua em uso

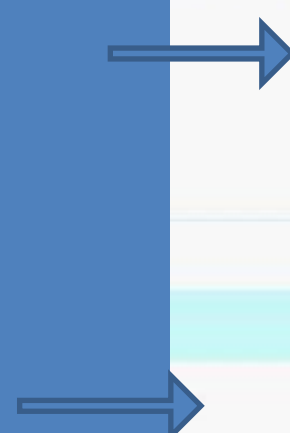
```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4  main()
5  {
6      int x;
7      setlocale(LC_ALL, "Portuguese");
8      for (x=1; x<=100; x++) {
9          printf("O valor de x = %d \n", x);
10     }
11     system ("pause");
12 }
```

For(**x= valor inicial**; **condição**; **incremento ou decremento**)

Estrutura de repetição - FOR

- O FOR também segue a mesma regra de IF. Caso haja mais de uma linha a ser executada pelo comando FOR, essas deverão estar agrupadas num bloco de dados ({ }).

```
for(int i=0; i<5; i++)  
{  
    cout<<"Digite um numero: ";  
    cin>>num;  
    if(num < menor) I  
    |  
} //fim do for
```



Estrutura de repetição - FOR

- Este comando não se limita a operações com constantes, por exemplo: $x=1$, executar enquanto x for menor ou igual a 100... Às vezes, o usuário pode entrar com dados para controlar o loop. Veja:
- No exemplo 2, a contagem começará a partir do número que o usuário digitar.

Estrutura de repetição – FOR

Exemplo 2

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4  main()
5  {
6      int x, numero;
7      setlocale(LC_ALL, "Portuguese");
8      printf("Digite um numero menor que 100 para iniciar a contagem : ");
9      scanf("%d", &numero);
10     for (x=numero; x<=100; x++) {
11         printf("O valor de x = %d \n", x);
12     }
13     system ("pause");
14 }
```

Estrutura de repetição - FOR

- No exemplo 3, a contagem começará de 1 e vai até o número que o usuário digitar.

Estrutura de repetição – FOR

Exemplo 3

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4  main()
5  {
6      int x, numero;
7      setlocale(LC_ALL, "Portuguese");
8      printf("Digite um numero menor que 100 para finalizar a contagem : ");
9      scanf("%d",&numero);
10     for (x=1; x<=numero; x++) {
11         printf("O valor de x = %d \n",x);
12     }
13     system ("pause");
14 }
```


Estrutura de repetição - FOR

- No exemplo 4, a contagem irá de 0 a 100 com intervalos determinados pelo usuário.

Estrutura de repetição – FOR

Exemplo 4

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4  main()
5  {
6      int x, numero;
7      setlocale(LC_ALL, "Portuguese");
8      printf("Digite qual será o intervalo da contagem : ");
9      scanf("%d", &numero);
10     for (x=1; x<=100; x+=numero) {
11         printf("O valor de x = %d \n", x);
12     }
13     system ("pause");
14 }
```

Estrutura de repetição – WHILE

- Uma outra forma de iteração (repetição) em C/C++ é o WHILE.
- *While* executa uma comparação com a variável. Se a comparação for verdadeira, ele executa o bloco de instruções ({ }).
- Procedemos da seguinte maneira:
 - WHILE (comparação)

Estrutura de repetição – WHILE

- Exemplo 1
- Programa que gera os números pares entre 0 e 100.

Estrutura de repetição – WHILE

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4  main()
5  {
6      int par;
7      setlocale(LC_ALL, "Portuguese");
8      printf("Os números pares de 0 a 100 são : \n");
9      par=0;
10     while (par<=100) {
11         printf(" %d ",par);
12         par+=2;
13     }
14     printf("\n\n\n\n");
15     system ("pause");
16 }
```

Estrutura de repetição – WHILE

- Algumas precauções
- O WHILE muitas vezes pode nos pregar peças. Se prestarmos atenção, no exemplo, inicializamos a variável par em 0, pois quando usamos no while uma variável sem inicializa-la pode causar comportamentos estranhos. E por que isso acontece?
- É simples. C/C++ **não inicializa variáveis automaticamente em 0**, ou seja, se você comparar uma variável não inicializada, essa variável pode ser qualquer coisa, ex.: 1557745, -9524778, 1 ou até mesmo 0.

Estrutura de repetição – WHILE

- Outro problema comum com o while é o loop infinito. O WHILE, diferentemente de FOR, não incrementa ou decrementa automaticamente uma variável, isso deve estar expressado dentro do bloco de instruções, como podemos ver na linha `par+=2`. Caso contrário, `par` sempre seria zero e nunca chegaria a 100 para o loop parar, causando o loop infinito.
- O ideal é utilizar o WHILE em um loop definido pelo usuário, que a partir de uma entrada ele termine. Ou seja, enquanto o usuário não fizer determinada ação, o loop continuará a dar voltas. Como no exemplo 2.

Estrutura de repetição – WHILE

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4  main()
5  {
6      int x;
7      setlocale(LC_ALL, "Portuguese");
8      x=1;
9      while (x!=0) {
10         printf("Digite o número 0 para finalizar : ");
11         scanf("%d",&x);
12         if (x>0) {
13             printf("Este número é positivo\n");
14         } else {
15             printf("Este número é negativo\n");
16         }
17     }
18     printf("\n\n\n\n");
19     system ("pause");
20 }
```


Estrutura de repetição – DO WHILE

- A estrutura de repetição DO WHILE parte do princípio de que deve-se fazer algo primeiro e só depois comparar uma variável para saber se o loop será executado mais uma vez.
- Devemos proceder da seguinte maneira:
 - **DO {bloco de instruções} WHILE (comparação);**

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int i;
    do {
        printf("Escolha uma opcao:\n");
        printf("(1) Opcao 1\n");
        printf("(2) Opcao 2\n");
        printf("(3) Opcao 3\n");
        scanf("%d", &i);

        }while((i < 1) || (i > 3));

    system("pause");
    return 0;
}
```

Estrutura de repetição – DO WHILE

- Usamos DO, depois escrevemos tudo que o DO deve fazer no bloco de instruções, no final do bloco colocamos o WHILE com a comparação entre parênteses, e não se esqueça que neste caso termina-se a linha do WHILE com ponto-e-vírgula (;). Veja o exemplo:

Estrutura de repetição – DO WHILE

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4  main()
5  {
6      int x;
7      setlocale(LC_ALL, "Portuguese");
8      do {
9          printf("Digite o número 0 para finalizar : ");
10         scanf("%d", &x);
11         if (x>0) {
12             printf("Este número é positivo\n");
13         } else {
14             printf("Este número é negativo\n");
15         }
16     } while (x!=0);
17     printf("\n\n\n\n");
18     system ("pause");
19 }
```

bibliografias

- SZWARCFITER, Jayme Luiz; MARKENZON, Lilian. Estruturas de dados e seus algoritmos. 2ed. Rio de Janeiro: LTC, 1994. 320p.
- TENENBAUM, Aaron M.; LANGSAM, Yedidyah; AUGENSTEIN, Moshé J.. Estruturas de dados usando C. São Paulo: Makron Books, 1995. 884p.
- VELOSO, Paulo et al.. Estruturas de dados. Rio de Janeiro: Campus, 2001. 228p.