UNIVERSIDADE
DO ESTADO DE MINAS GERAIS | UEMG
UNIDADE PASSOS

Estruturas de Dados 1
Revisão

Mailson de Queiroz Proença

Estrutura básica de um programa C++

```
#include <iostream>

using namespace std;

int main ()
{
  cout << "Olá mundo!";
  return 0;
}
```

Entrada de dados e comentários no código

```
// Este é um comentário de uma linha

/*
  Este é  um comentário
  de várias
  linhas
*/

#include <iostream>

using namespace std;

int main ()
{
  int x;
  cout << "Digite um número: ";
  cin >> x;
  cout << "\nVocê digitou o número: " << x << endl;
  return 0;
}
```

Os diferentes tipos de variáveis

- **bool**: Em geral, utiliza **1** byte da memória, valores: **true** ou **false**.
- **char**: Em geral, utiliza **1** byte da memória, permite armazenar um caractere.
- **string**:
- **int**: Em geral, utiliza **2** bites da memória, valores: de **-32 768** a **32 767.**
- **float**: Em geral, utiliza **4** bites da memória, valores: de **1.2e-308** a **3.4e-38**.
- **double**: Em geral, utiliza **8** bites da memória, valores: de **2.2e-308** a **3.4e-38.**

Operadores aritméticos

```
#include <iostream>
using namespace std;

int main() {

int soma = 5 + 5; // o operador '+' realiza somas.
double subtracao = 5 - 5; // o operador '-' efetua subtração.
float multiplicacao = 5.1 * 0.5; // o operador '*' efetua multiplicação.
int divisao = 100 / 2; // o operador '/' efetua divisão.
int modulo = 51 % 5; // retorna o resto da divisão inteira.

cout << "Resultados: " << soma << ", " << subtracao << ", " <<
multiplicacao << ", " << divisao << ", " << modulo << endl;

}
```

Estruturas Condicionais

- **if**:
  - Estrutura Condicional Simples;
  - Estrutura Condicional Composta;
- **switch**

Operadores Condicionais ou Relacionais

- Os operadores condicionais ou relacionais são operadores binários que devolvem os valores lógicos verdadeiro e falso.

OPERADOR	DESCRIÇÃO	EXEMPLO
==	Igual a	A == B
>	Maior que	A > B
<	Menor que	A < B
>=	Maior ou igual a	A >= B
<=	Menor ou igual a	A <= B
!=	Diferente de	A != B

Operadores Lógicos

- Os principais operadores lógicos são: && e ||.

&&

CONDIÇÃO 1	CONDIÇÃO 2	RESULTADO
V	V	V
V	F	F
F	V	F
F	F	F

||

CONDIÇÃO 1	CONDIÇÃO 2	RESULTADO
V	V	V
V	F	V
F	V	V
F	F	F

Operadores Lógicos

```
if(x == 3)
      cout << "Número igual a 3";

if(x > 5 && x<10)
      cout << "Número entre 5 e 10";

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

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

Switch

- Outra forma de estrutura seletiva;
- É quase que um if com várias possibilidades, mas com algumas diferenças importantes:
  - Os cases não aceitam operadores lógicos. Portanto, não é possível fazer uma comparação. Isso limita o case a apenas valores definidos.
  - O switch executa seu bloco em cascata. Ou seja, se a variável indicar para o primeiro case e dentro do switch tiver 5 cases, o switch executará todos os outros 4 cases a não ser que utilizemos o comando para sair do switch.

Switch

```
SWITCH (variável){
    CASE valor1:
        Dados a serem executados
        BREAK;
    CASE valor2:
        Dados a serem executados
        BREAK;
}
```

Switch

```
#include <iostream>
using namespace std;

int main ()
{
	int a = 1, b = 2, operacao, c;
	cout << "Que operacao deseja realizar?\n 1. Adicao\t2.Subtracao\t3.Multiplicacao\t4.Divisao\n\n=>";
	cin >> operacao;
	switch (operacao)
	{
		case 1:
			c = a + b;
			cout << a << " + " << b << " = " << c << "\n\n";
			break;
	case 2:
			c = a - b;
			cout << a << " - " << b << " = " << c << "\n\n";
			break;
	case 3:
			c = a * b;
			cout << a << " * " << b << " = " << c << "\n\n";
			break;
	case 4:
			c = a / b;
			cout << a << " / " << b << " = " << c << "\n\n";
			break;
	}
	return 0;
}
```

Switch - Default

```
#include <iostream>
using namespace std;

int main ()
{
	int a = 1, b = 2, operacao,c;
	cout << "Que operacao deseja realizar?\n 1. Adicao\t2.Subtracao\t3.Multiplicacao\t4.Divisao\n\n=>";
	cin >> operacao;
	switch (operacao)
	{
		case 1:
			c = a + b;
			cout << a << " + " << b << " = " << c << "\n\n";
			break;
	case 2:
			c = a - b;
			cout << a << " - " << b << " = " << c << "\n\n";
			break;
	case 3:
			c = a * b;
			cout << a << " * " << b << " = " << c << "\n\n";
			break;
	case 4:
			c = a / b;
			cout << a << " / " << b << " = " << c << "\n\n";
			break;
	default:
			cout << "Opcao invalida!";
	}
	return 0;
}
```

If ternário ou condicional “?”

- Sintaxe:

```
condição ? Código se verdadeiro : Código se falso
```

- Ou seja, este operador testa a expressão relacional e se o resultado for verdadeiro executa logo a 1ª afirmação caso contrário executa a segunda.
- ou seja isto não é mais do que um if-else.

If ternário ou condicional “?”

```
#include <iostream>

using namespace std;

int main()
{
    int num;
    cout << “Digite um numero: ";
    cin >> num;
    cout << “O numero e " << (num % 2 == 0 ? “par" : “impar") << endl;
    return 0;
}
```

Estruturas de Repetição

- for
- while
- do while

While

- É um laço que ordena o computador a executar determinadas instruções enquanto uma condição for verdadeira;
- Comando que leve a condição de execução a ser falsa em algum momento

While

```
#include <iostream>

using namespace std;

int main()
{
  int contador;                 // Declara a variável contador.
  contador=1;                   // contador recebe o valor 1.
  while (contador<=10)          // Enquanto contador for menor ou igual a 10.
  {
    cout << contador << endl; // Imprime contador.
    contador++;                 // Incrementa contador em uma unidade.
  }
  return 0;
}
```

Do-while

- O laço *do-while* é um *while* invertido, onde você coloca as instruções a serem repetidas antes da verificação da condição de execução. Isso significa que os comandos do laço serão executados ao menos uma vez.

Do-while

```
#include <iostream>

using namespace std;

int main()
{
  int contador;                  // Declara a variável contador.
  contador=1;                    // contador recebe o valor 1.
  do {
    cout << contador << endl; // Imprime contador.
    contador++;                  // Incrementa contador em uma unidade.
  } while (contador<=10);      // Enquanto contador for menor ou igual a 10.
  return 0;
}
```

for

- Como o uso de uma variável como contador nos laços é algo frequente, foi criado um outro laço que traz em sua estrutura campos para abrigar os comandos de atribuição de valor inicial e incremento/decremento do contador.

for

```
#include <iostream>

using namespace std;

int main()
{
  int contador;                                  // Declara a variável contador.
  for (contador=1; contador<=10; contador++) // Inicia o laço.
    cout << contador << endl;                    // Imprime contador.
  return 0;
}
```

for

- É importante deixar claro que nenhum dos três parâmetros utilizados no laço *for* é obrigatório.
- Caso não haja necessidade de utilizar um ou mais deles, basta deixar seu espaço em branco.

```
#include <iostream>

using namespace std;

int main()
{
  for (;;)
    cout << "Eu sou um laço infinito." << endl;
  return 0;
}
```

O Comando break

- O que o break faz é quebrar a execução para fora do bloco de código onde ele está presente:

O Comando continue

- Esta instrução é bem parecida com o break, mas algo diferente.
- Pois em vez de mandar a execução para fora do bloco manda-a para a avaliação do loop.
- Ou seja faz saltar uma determinada iteração do loop, enquanto o break faz acabar o loop

Incrementar/decrementar

- a=a+1 é equivalente a ter a+=1;
- Podemos ter ++a ou ainda a++. Eles são parecidos mas diferentes, é a questão do **prefixo** e **pós-fixo**. A diferença é que:
  - O prefixo, faz o incremento ainda durante a instrução;
  - O pós-fixo faz o incremento quando se passa para a instrução seguinte.

Incrementar/decrementar

```
#include <iostream>
 using namespace std;
 int main(void)
 {
   int num = 2;
   cout << num << ”\n”;
   cout << ++num << ”\n”;
   cout << num++ <<”\n”;
   cout << num << ”\n”;
   return 0;
 }
```

Criação de Menu

```
#include <iostream>
 using namespace std;
 int main ()
 {
   int  i;
   do
    {
           cout << "\n\nEscolha a fruta pelo numero:\n\n";
           cout << "\t(1)...Mamao\n";
           cout << "\t(2)...Abacaxi\n";
           cout << "\t(3)...Laranja\n\n";
           cin >> i;
    } while ((i<1)||(i>3));
   switch (i)
       {
           case 1:
                      cout << ("\t\tVoce escolheu Mamao.\n");
                 break;
           case 2:
                      cout <<"\t\tVoce escolheu Abacaxi.\n";
                      break;
           case 3:
                      cout << ("\t\tVoce escolheu Laranja.\n");
                      break;
        }
   return(0);
 }
```

Strings

```
char palavra[20];

string palavra;
```

Strings – funções úteis

- As seguintes funções estão no cabeçalho da biblioteca **<cctype>:**
  - **toupper()** – (to+upper) retorna a maiúscula de uma letra. é uma função de um argumento – o caractere. no caso do argumento não ser uma letra, a função retorna o mesmo caractere que é argumento.
  - **tolower()** – (to+lower) o mesmo comportamento que toupper(), porém com o resultado em minúscula.

Strings – funções úteis

- Funções que verificam o caractere. Estas funções recebem apenas um argumento, o caractere e retornam um valor booleano:

Função	Descrição
isalpha	Retorna verdadeiro se o argumento é uma letra do alfabeto; falso em caso contrário.
isalnum	Retorna verdadeiro se o argumento é uma letra do alfabeto ou um dígito; falso em caso contrário.
isdigit	Retorna verdadeiro se o argumento é um dígito; falso em caso contrário.
islower	Retorna verdadeiro se o argumento é uma letra minúscula, falso em caso contrário.
isprint	Retorna verdadeiro se o argumento é um caractere imprimível (incluíndo espaços); falso em caso contrário.
ispunct	Retorna verdadeiro se o argumento é um sinal de pontuação (caracteres imprimíveis que não sejam letras, dígitos ou espaço); falso em caso contrário.
isupper	Retorna verdadeiro se o argumento é uma letra maiúscula; falso em caso contrário.
isspace	Retorna verdadeiro se o argumento é um espaço, tabulação ou nova linha; falso em caso contrário.

Exercícios

- Faça um programa que leia dois valores para as variáveis A e B e efetue a troca dos valores de forma que a variável A passe a possuir o valor da variável B e a variável B passe a possuir o valor da variável A. Apresente os valores trocados.

Exercícios

O programa de uma loja de móveis mostra o seguinte menu na tela de vendas:

1-Venda a Vista

2-Venda a Prazo 30 dias

3-Venda a Prazo 60 dias

4-Venda a Prazo com 90 dias

5-Venda com cartão de débito

6-Venda com cartão de crédito

Escolha a opção:

Faça um programa que receba o valor da venda, escolha a condição de pagamento no menu e mostre o total da venda final conforme condições a seguir:

Venda a Vista - desconto de 10%

Venda a Prazo 30 dias - desconto de 5%

Venda a Prazo 60 dias - mesmo preço

Venda a Prazo 90 dias - acréscimo de 5%

Venda com cartão de débito - desconto de 8%

Venda com cartão de crédito - desconto de 7%

Exercícios

O cardápio de uma lanchonete é o seguinte:

Especificação	Preço unitário
100 Cachorro quente	1,10
101 Bauru simples	1,30
102 Bauru c/ovo	1,50
103 Hamburger	1,10
104 Cheeseburger	1,30
105 Refrigerante	1,00

Desenvolva um programa que leia o código do item pedido, a quantidade e calcule o valor a ser pago por aquele lanche. Considere que a cada execução somente será calculado um item.

Exercícios

- Crie um programa que calcule o fatorial de um número digitado pelo usuário:

Exercícios

- Construa um Algoritmo que, para um grupo de 50 valores inteiros, determine:

a) A soma dos números positivos;

b) A quantidade de valores negativos;

Exercícios

- Faça um algoritmo que imprima os múltiplos positivos de 7, inferiores a 1000.

Exercícios

- Faça um algoritmo que:

a) Leia o nome;

b) Leia o sobrenome;

c) Concatene o nome com o sobrenome;

d) Apresente o nome completo.

Exercícios

- Suponha que uma escola utilize, como código de matrícula, um número inteiro no formato AASDDD, onde:
  - Os dois primeiros dígitos, representados pela letra A, são os dois últimos algarismos do ano da matrícula;
  - O terceiro dígito, representado pela letra S, vale 1 ou 2, conforme o aluno tenha se matriculado no 1º ou 2º semestre;
  - Os quatro últimos dígitos, representados pela letra D, correspondem à ordem da matrícula do aluno, no semestre e no ano em questão.
- Crie um algoritmo que leia o número de matrícula de um aluno e imprima o ano e o semestre em que ele foi matriculado.