DISCIPLINA: ESTRUTURAS DE DADOS (4 AULAS SEMANAIS)

OBJETIVOS

Criar e manipular tipos abstratos de dados: listas, pilhas, filas e árvores.

EMENTA

Pilhas, filas, alocação dinâmica, recursividade, listas encadeadas, tabelas de espalhamento e árvores.

Relembrando a linguagem C:

In [1]:
```
%%file aula01/ex01.c
#include <stdio.h>

int main() {
    printf("Ola mundo 01!");
    return 0;
}
```

```
Overwriting aula01/ex01.c
```

In [2]:
```
!gcc aula01/ex01.c -o aula01/ex01
```

In [3]:
```
!./aula01/ex01
```

```
Ola mundo 01!
```

In [4]:
```
%%file aula01/ex02.cpp
#include <iostream>
using namespace std;

int main() {
    cout << "Ola mundo 02!";
    return 0;
}
```

```
Overwriting aula01/ex02.cpp
```

In [5]:
```
!g++ aula01/ex02.cpp -o aula01/ex02
```

In [6]:
```
!./aula01/ex02
```

```
Ola mundo 02!
```

TIPOS DE DADOS

- char : caracter

- string: conjuntos de caracteres
- int : inteiros
- float : precisão simples (32-bits aproximadamente sete digitos)
- double : precisão dupla (64-bits entre 15 e 16 digitos)
- bool : booleano

É importante ressaltar que embora C++ disponha do tipo bool, qualquer valor diferente de zero é interpretado como sendo verdadeiro (true). O valor zero é interpretado como sendo falso (false). O exemplo abaixo cria variáveis dos tipos básicos e exibe seus valores.

In [7]:
```
%%file aula01/ex04.cpp
#include <iostream>
using namespace std;

int main() {
    char charVar;
    int intVar = 298;
    float floatVar = 49.95;
    double doubleVar = 99.9999;
    bool boolVar = (2 > 3);

    charVar = 't';

    cout << "charVar = "
         << charVar
         << "\n";
    cout << "intVar = "
         << intVar
         << "\n";
    cout << "floatVar = "
         << floatVar
         << "\n";
    cout << "doubleVar = "
         << doubleVar
         << "\n";
    cout << "boolVar = "
         << boolVar
         << "\n";
    return 0;
}
```

```
Overwriting aula01/ex04.cpp
```

In [8]:
```
!g++ aula01/ex04.cpp -o aula01/ex04
```

In [9]:
```
!./aula01/ex04
```

```
charVar = t
intVar = 298
floatVar = 49.95
doubleVar = 99.9999
boolVar = 0
```

Escape

- \n caractere de nova linha
- \t caractere de tabulação (tab)
- \b caractere backspace

- \" aspa dupla
- \' aspa simples
- \? ponto de interrogação
- \ barra invertida

In [10]:
```
%%file aula01/ex05.cpp
#include <iostream>
using namespace std;

int main() {
    cout << "\"Frase entre aspas\"'\n";
    cout << "Alguma \tduvida\?\n\\";
    return 0;
}
```

```
Overwriting aula01/ex05.cpp
```

In [11]:
```
!g++ aula01/ex05.cpp -o aula01/ex05
```

In [12]:
```
!./aula01/ex05
```

```
"Frase entre aspas"'
Alguma  duvida?
\
```

Variáveis

In [13]:
```
%%file aula01/ex06.cpp
#include <iostream>
using namespace std;

int main() {
    cout << "*** Tamanhos das variaveis ***\n";
    cout << "Tamanho de char =\t"
         << sizeof(char)
         << " bytes.\n";
    cout << "Tamanho de int =\t"
         << sizeof(int)
         << " bytes.\n";
    cout << "Tamanho de float =\t"
         << sizeof(float)
         << " bytes.\n";
    cout << "Tamanho de double =\t"
         << sizeof(double)
         << " bytes.\n";
    cout << "Tamanho de bool =\t"
         << sizeof(bool)
         << " bytes.\n";
    return 0;
}
```

```
Overwriting aula01/ex06.cpp
```

In [14]:
```
!g++ aula01/ex06.cpp -o aula01/ex06
```

In [15]:
```
!./aula01/ex06
```

```
*** Tamanhos das variaveis ***
```

```
Tamanho de char =       1 bytes.
Tamanho de int =        4 bytes.
Tamanho de float =      4 bytes.
Tamanho de double =     8 bytes.
Tamanho de bool =       1 bytes.
```

In [16]:

```
%%file aula01/ex07.cpp
#include <iostream>
using namespace std;

int main() {
    int largura = 7, comprimento;
    comprimento = 8;
    int area = largura * comprimento;
    cout << "*** Valores finais ***\n";
    cout << "Largura = "
        << largura << "\n";
    cout << "Comprimento = "
        << comprimento << "\n";
    cout << "Area = "
        << area << "\n";
    return 0;
}
```

```
Overwriting aula01/ex07.cpp
```

In [17]:

```
!g++ aula01/ex07.cpp -o aula01/ex07
```

In [18]:

```
!./aula01/ex07
```

```
*** Valores finais ***
Largura = 7
Comprimento = 8
Area = 56
```

Variáveis Unsigned

In [19]:

```
%%file aula01/ex08.cpp
#include <iostream>
#include <iomanip>
#include <math.h>
using namespace std;

int main() {
    cout.setf(ios::fixed);
    cout << "Tipo\t\t\tTamanho\t\tValores\n";

    cout <<"short int: \t\t"
        << sizeof(short int)
        << " bytes\t\t"
        << setprecision(0) << pow(2,8*sizeof(unsigned short int))/2*-1
        << " a "
        << pow(2,8*sizeof(unsigned short int))/2-1
        << "\n";

    cout << "unsigned short int: \t"
        << sizeof(unsigned short int)
        << " bytes\t\t"
        << 0
        << " a "
        << pow(2,8*sizeof(unsigned short int))-1
        << "\n";
```

```
    cout <<"int: \t\t\t"
         << sizeof(int)
         << " bytes\t\t"
         << pow(2,8*sizeof(int))/2*-1
         << " a "
         << pow(2,8*sizeof(int))/2-1
         << "\n";


    cout << "unsigned int: \t\t"
         << sizeof(unsigned int)
         << " bytes\t\t"
         << 0
         << " a "
         << pow(2,8*sizeof(int))-1
         << "\n";

    cout <<"long int: \t\t"
         << sizeof(long int)
         << " bytes\t\t"
         << pow(2,8*sizeof(long int))/2*-1
         << " a "
         << (pow(2,8*sizeof(long int))/2)-1
         << "\n";

    cout << "unsigned long int: \t"
         << sizeof(unsigned long int)
         << " bytes\t\t"
         << 0
         << " a "
         << pow(2,8*sizeof(unsigned long int))-1
         << "\n";
    return 0;
}
```

```
Overwriting aula01/ex08.cpp
```

In [20]:

```
!g++ aula01/ex08.cpp -o aula01/ex08
```

In [21]:

```
!./aula01/ex08
```

```
Tipo                    Tamanho         Valores
short int:              2 bytes         -32768 a 32767
unsigned short int:     2 bytes         0 a 65535
int:                    4 bytes         -2147483648 a 2147483647
unsigned int:           4 bytes         0 a 4294967295
long int:               8 bytes         -9223372036854775808 a 9223372036854775808
unsigned long int:      8 bytes         0 a 18446744073709551616
```

In [22]:

```
%%file aula01/ex09.cpp
#include <iostream>
using namespace std;

int main() {
    unsigned short int usVar;
    usVar = 65535;
    cout << "Valor inicial = " << usVar << "\n";
    usVar = usVar + 1;
    cout << "Somando 1 = " << usVar << "\n";
    usVar = usVar + 1;
    cout << "Somando mais 1 = " << usVar << "\n";
```

```
    return 0;
}
```

```
Overwriting aula01/ex09.cpp
```

In [23]:
```
!g++ aula01/ex09.cpp -o aula01/ex09
```

In [24]:
```
!./aula01/ex09
```

```
Valor inicial = 65535
Somando 1 = 0
Somando mais 1 = 1
```

Strings

In [25]:
```
%%file aula01/ex10.cpp
#include <iostream>
using namespace std;

int main() {
    string s1 = "Ola";
    string s2 = "Mundo";
    cout << s1 + " " + s2;
    return 0;
}
```

```
Overwriting aula01/ex10.cpp
```

In [26]:
```
!g++ aula01/ex10.cpp -o aula01/ex10
```

In [27]:
```
!./aula01/ex10
```

```
Ola Mundo
```

Funções

In [28]:
```
%%file aula01/ex11.cpp
#include <iostream>
using namespace std;

void digaAlo() {
    cout << "Alo, Mundo!";
}

int main() {
    digaAlo();
    return 0;
}
```

```
Overwriting aula01/ex11.cpp
```

In [29]:
```
!g++ aula01/ex11.cpp -o aula01/ex11
```

In [30]:
```
!./aula01/ex11
```

```
Alo, Mundo!
```

Funções: parametros e protótipo de função

In [31]:

```
%%file aula01/ex12.cpp
#include <iostream>
using namespace std;

void digaAlo(string nome, int idade, float salario);

int main() {
    string nome;
    int idade;
    float salario;

    cout << "Digite o nome da pessoa:";
    cin >> nome;
    cout << "\nIdade:";
    cin >> idade;
    cout << "\nSalario:";
    cin >> salario;

    digaAlo("Bob", 44, 23.5);
    digaAlo("Patrick", 55, 88);
    return 0;
}

void digaAlo() {
    cout << "nome: " << nome << "\n";
    cout << "idade: " << idade << "\n";
    cout << "salario: " << salario << "\n";
}
```

```
Overwriting aula01/ex12.cpp
```

In [32]:

```
%%file aula01/ex12.cpp
#include <iostream>
using namespace std;


int digaAlo(string nome, int &idade, float salario);

int main() {
    string nomeBob;
    int idadeBob, idadePatrick;
    float salarioBob, salarioPatrick=0;

    cout << "Digite o nome da pessoa:";
    nomeBob = "Bob"; //cin >> nomeBob;

    cout << "\nIdade:";
    idadeBob = 77; //cin >> idadeBob;

    cout << "\nSalario:";
    salarioBob = 77.5; //cin >> salarioBob;


    cout << "\n---Main--\n";
    cout << "nome: " << nomeBob << "\n";
    cout << "idade: " << idadeBob << "\n";
    cout << "salario: " << salarioBob << "\n";
    cout << "---Main--\n";

    digaAlo(nomeBob, idadeBob, salarioBob);

    cout << "\n---Main--\n";
```

```
    cout << "nome: " << nomeBob << "\n";
    cout << "idade: " << idadeBob << "\n";
    cout << "salario: " << salarioBob << "\n";
    cout << "---Main--\n";

    idadePatrick = 12;
    salarioPatrick = 15;
    cout << "\n---Main--\n";
    cout << "idadePatrick: " << idadePatrick << "\n";
    cout << "salarioPatrick: " << salarioPatrick << "\n";
    cout << "---Main--\n";

    // digaAlo("Patrick", 12 , 669.43); --> não pode
    salarioPatrick = digaAlo("Patrick", idadePatrick, 669.43);

    cout << "\n---Main--\n";
    cout << "idadePatrick: " << idadePatrick << "\n";
    cout << "salarioPatrick: " << salarioPatrick << "\n";
    cout << "---Main--\n";

    return 0;
}

int digaAlo(string nome, int &idade, float salario) {
    cout << "\n---DigaAlo--\n";
    cout << "nome: " << nome << "\n";
    idade = 8;
    salario = 1000;
    cout << "idade: " << idade << "\n";
    cout << "salario: " << salario << "\n";
    cout << "---DigaAlo--\n";
    return 500;
}
```

```
Overwriting aula01/ex12.cpp
```

In [33]:

```
!g++ aula01/ex12.cpp -o aula01/ex12
```

In [34]:

```
!./aula01/ex12
```

```
Digite o nome da pessoa:
Idade:
Salario:
---Main--
nome: Bob
idade: 77
salario: 77.5
---Main--

---DigaAlo--
nome: Bob
idade: 8
salario: 1000
---DigaAlo--

---Main--
nome: Bob
idade: 8
salario: 77.5
---Main--

---Main--
idadePatrick: 12
salarioPatrick: 15
```

```
---Main--

---DigaAlo--
nome: Patrick
idade: 8
salario: 1000
---DigaAlo--

---Main--
idadePatrick: 8
salarioPatrick: 500
---Main--
```

Entrada de dados

In [35]:

```
%%file aula01/ex13.cpp
#include <iostream>
using namespace std;

int Soma(int i, int j) {
    int k;
    cout << "Estamos na funcao Soma().\n";
    cout << "Valores recebidos: \n";
    cout << "i = "
         << i
         << ", j = "
         << j
         << "\n";
    k = i + j;
    return (k);
}

int main() {
    int x, y, z;
    cout << "Estamos em main()\n";
    cout << "\nDigite o primeiro num. + <Enter>";
    cin >> x;
    cout << "\nDigite o segundo num. + <Enter>";
    cin >> y;
    cout << "Chamando funcao Soma()...\n";
    z = Soma(x, y);
    cout << "Voltamos a main()\n";
    cout << "Novo valor de z = "
         << z
         << "\n";
    return 0;
}
```

```
Overwriting aula01/ex13.cpp
```

In [36]:

```
!g++ aula01/ex13.cpp -o aula01/ex13
#!./aula01/ex13
```

Operadores Matemáticos

- + adição
- - subtração
- * multiplicação
- / divisão
- % módulo

In [37]:
```
%%file aula01/ex14.cpp
#include <iostream>
using namespace std;

int main()
{
    cout << "*** Resto da divisao inteira ***\n";
    cout << "40 % 4 = "
        << 40 % 4
        << "\n";
    cout << "41 % 4 = "
        << 41 % 4
        << "\n";
    cout << "42 % 4 = "
        << 42 % 4
        << "\n";
    cout << "43 % 4 = "
        << 43 % 4
        << "\n";
    cout << "44 % 4 = "
        << 44 % 4
        << "\n";
    return 0;
} // Fim de main()
```

```
Overwriting aula01/ex14.cpp
```

In [38]:
```
!g++ aula01/ex14.cpp -o aula01/ex14
!./aula01/ex14
```

```
*** Resto da divisao inteira ***
40 % 4 = 0
41 % 4 = 1
42 % 4 = 2
43 % 4 = 3
44 % 4 = 0
```

In [39]:
```
%%file aula01/ex15.cpp
#include <iostream>
using namespace std;

int main() {
    short unsigned int diferenca;
    short unsigned  int numMaior = 1000;
    short unsigned  int numMenor = 300;
    cout << "\nnumMaior = "
        << numMaior
        << ", numMenor = "
        << numMenor
        << "\n";
    diferenca = numMaior - numMenor;
    cout << "\nnumMaior - numMenor = "
        << diferenca
        << "\n";
    diferenca = numMenor - numMaior;
    cout << "\nnumMenor - numMaior = "
        << diferenca
        << "\n";
    return 0;
}
```

```
Overwriting aula01/ex15.cpp
```

In [40]:

```
!g++ aula01/ex15.cpp -o aula01/ex15
!./aula01/ex15
```

```
numMaior = 1000, numMenor = 300

numMaior - numMenor = 700

numMenor - numMaior = 64836
```

Operadores incremento / decremento

In [41]:

```
%%file aula01/ex16.cpp
#include <iostream>
using namespace std;

int main() {
    int a, b, x, y, d, e;
    d++;
    e++;
    cout << "d: " << e;
    a = b = 10;
    x = a++;
    y = ++b;
    a = a + 1;
    a++;
    cout << "a: "
         << a
         << "\nb: "
         << b
         << "\nx: "
         << x
         << "\ny: "
         << y;
    cout << "\n\nDecremento:\n";
    x = a--;
    y = --b;
    cout << "a: "
         << a
         << "\nb: "
         << b
         << "\nx: "
         << x
         << "\ny: "
         << y;
    return 0;
}
```

```
Overwriting aula01/ex16.cpp
```

In [42]:

```
!g++ aula01/ex16.cpp -o aula01/ex16
```

In [43]:

```
!./aula01/ex16
```

```
d: 1a: 13
b: 11
x: 10
y: 11

Decremento:
a: 12
b: 10
```

```
x: 13
y: 10
```

Estrutura condicional

- argc – é um valor inteiro que indica a quantidade de argumentos que foram passados ao chamar o programa.
- argv – é um vetor de char que contém os argumentos, um para cada string passada na linha de comando.
- argv[0] armazena o nome do programa que foi chamado no prompt, sendo assim, argc é pelo menos igual a 1, pois no mínimo existirá um argumento.
- int atoi (const char * str); : converte string to integer

In [44]:
```
%%file aula01/ex17.cpp
#include <iostream>
using namespace std;

int main(int argc, char *argv[ ]) {
    int a;
    a = atoi(argv[1]);
    cout << argv[0] << "\n";

    if (a > 10) {
        cout << "a: " << a << " maior que 10!\n";
    } else {
        cout << "a: " << a <<  " menor que 10!\n";
        if (a > 5) {
            cout << "a: " << a << " maior que 5\n";
        }
    }
    cout << "\n";
    return 0;
}
```

```
Overwriting aula01/ex17.cpp
```

In [45]:
```
!g++ aula01/ex17.cpp -o aula01/ex17
```

In [46]:
```
!./aula01/ex17 11
```

```
./aula01/ex17
a: 11 maior que 10!
```

In [47]:
```
!./aula01/ex17 9
```

```
./aula01/ex17
a: 9 menor que 10!
a: 9 maior que 5
```

In [48]:
```
!./aula01/ex17 4
```

```
./aula01/ex17
a: 4 menor que 10!
```

Operadores lógicos

Operador	Símbolo	Exemplo
AND	&&	expressao1 && expressao2
OR	||	expressao1 || expressao2
NOT	!	!expressao

In [49]:
```
%%file aula01/ex18.cpp
#include <iostream>
using namespace std;

int main(int argc, char *argv[ ]) {
    int a;
    a = atoi(argv[1]);

    if ((a > 10) && (a < 20)) {
        cout << "a: " << a << " maior que 10 e menor que 20!\n";
    }

    if ((a <= 10) || (a >= 20)) {
        cout << "a: " << a << " menor que 10 ou maior que 20!\n";
    }

    if (! (a == 11) ) {
        cout << "a: " << a << " diferente de 11!\n";
    }

    if (a != 11) {
        cout << "a: " << a << " diferente de 11!\n";
    }

    return 0;
}
```

```
Overwriting aula01/ex18.cpp
```

In [50]:
```
!g++ aula01/ex18.cpp -o aula01/ex18
```

In [51]:
```
!./aula01/ex18 11
```

```
a: 11 maior que 10 e menor que 20!
```

In [52]:
```
!./aula01/ex18 9
```

```
a: 9 menor que 10 ou maior que 20!
a: 9 diferente de 11!
a: 9 diferente de 11!
```

In [53]:
```
!./aula01/ex18 21
```

```
a: 21 menor que 10 ou maior que 20!
a: 21 diferente de 11!
a: 21 diferente de 11!
```

Operador condicional ternário

(expressao) ? (<valor-se-verdadeiro>) : (<valor-se-falso>);

Esta operação pode ser interpretada da seguinte forma: se expressao1 for verdadeira, retorne o valor de expressao2; caso contrario, retorne o valor de expressao3.

In [54]:
```
%%file aula01/ex19.cpp
#include <iostream>
using namespace std;

int main(int argc, char *argv[ ]) {
    int a;
    string resultado;

    a = atoi(argv[1]);
    resultado = (a > 10) ? ("maior") : ("menor");
    cout << "a: " << a << " resultado: " << resultado << "\n\n";
    return 0;
}
```

```
Overwriting aula01/ex19.cpp
```

In [55]:
```
!g++ aula01/ex19.cpp -o aula01/ex19
```

In [56]:
```
!./aula01/ex19 11
```

```
a: 11 resultado: maior
```

In [57]:
```
!./aula01/ex19 9
```

```
a: 9 resultado: menor
```

In [58]:
```
!./aula01/ex19 21
```

```
a: 21 resultado: maior
```

In [59]:
```
!cd aula01 && ./ex19 8
```

```
a: 8 resultado: menor
```

Variáveis Locais e Globais

In [60]:
```
%%file aula01/ex20.cpp
#include <iostream>
using namespace std;

int num(int b);
int aGlobal = 4;
#define MAX 4

int main() {
    int a = 3;
    int b = 1;
    int c = 2;
    cout << "Max:" << MAX << "\n";
    cout << "aGlobal: " << aGlobal << "\n";
    cout << "a main: " << a << "\n";
    cout << "b main: " << b << "\n";
```

```
    cout << "c main: " << c << "\n";
    c = num(4);
    cout << "aGlobal: " << aGlobal << "\n";
    cout << "a main: " << a << "\n";
    cout << "b main: " << b << "\n";
    cout << "c main: " << c << "\n";
    return 0;
}

int num(int b) {
    int a = 5;
    cout << "\n---inicio função---\n";
    cout << "a na função: " << a << "\n";
    cout << "b na função: " << b << "\n";
    cout << "aGlobal na função: " << aGlobal << "\n";
    aGlobal++;
    b++;
    cout << "b na função pós incremento: " << b << "\n";
    cout << "---fim função---\n\n";
    return b;
}
```

```
Overwriting aula01/ex20.cpp
```

In [61]:

```
ex='ex20'
!g++ aula01/"$ex".cpp -o aula01/$ex
```

In [62]:

```
!./aula01/$ex
```

```
Max:4
aGlobal: 4
a main: 3
b main: 1
c main: 2

---inicio função---
a na função: 5
b na função: 4
aGlobal na função: 4
b na função pós incremento: 5
---fim função---

aGlobal: 5
a main: 3
b main: 1
c main: 5
```

Estruturas de Repetição

While

In [63]:

```
%%file aula01/ex21.cpp
#include <iostream>
using namespace std;

int main() {
    int contador = 0;
    while(contador < 3) {
        contador++;
        cout << "\nContador = " << contador;
    }
    cout << "\n\nValor final: Contador = " << contador;
```

```
    return 0;
}
```

```
Overwriting aula01/ex21.cpp
```

In [64]:
```
ex='ex21'
!g++ aula01/"$ex".cpp -o aula01/$ex
```

In [65]:
```
!./aula01/$ex
```

```
Contador = 1
Contador = 2
Contador = 3

Valor final: Contador = 3
```

continue e break

In [66]:
```
%%file aula01/ex22.cpp
#include <iostream>
using namespace std;

int main() {
    int contador = 0;
    while(true) {
         contador++;
         if (contador > 20)
             break;
         if (contador > 10) {
             continue;
         }
         cout << "Contador = " << contador << "\n";
    }
    cout << "\n\nValor final = " << contador;
    return 0;
}
```

```
Overwriting aula01/ex22.cpp
```

In [67]:
```
ex='ex22'
!g++ aula01/"$ex".cpp -o aula01/$ex
!./aula01/$ex
```

```
Contador = 1
Contador = 2
Contador = 3
Contador = 4
Contador = 5
Contador = 6
Contador = 7
Contador = 8
Contador = 9
Contador = 10


Valor final = 21
```

do while

Recomendado quando eu quero garantir que o bloco da estrutura de repetição seja executado pelo menos uma vez.

In [68]:
```
%%file aula01/ex23.cpp
#include <iostream>
using namespace std;

int main() {
    int contador = 11;
    do {
        contador++;
        cout << "\nContador = " << contador;
    } while(contador < 10);
    cout << "\n\nValor final: Contador = " << contador;
    return 0;
}
```

```
Overwriting aula01/ex23.cpp
```

In [69]:
```
ex='ex23'
!g++ aula01/"$ex".cpp -o aula01/$ex
!./aula01/$ex
```

```
Contador = 12

Valor final: Contador = 12
```

In [70]:
```
%%file aula01/ex23b.cpp
#include <iostream>
using namespace std;

int main() {
    int a;
    do {
        cout << "Digite um valor para a (igual ou maior que 10): ";
        cin >> a;
        cout << "\na = " << a << "\n";
    } while(a < 10);
    cout << "\n\nValor final: a = " << a;
    return 0;
}
```

```
Overwriting aula01/ex23b.cpp
```

In [71]:
```
ex='ex23b'
!g++ aula01/"$ex".cpp -o aula01/$ex
```

for

Recomendado quando eu sei antecipadamente quantas vezes eu quero repetir um bloco de instruções.

In [72]:
```
%%file aula01/ex24.cpp
#include <iostream>
using namespace std;

int main() {
    int contador;

    contador = 0;
    while (contador <= 10) {
        cout << "Contador (while) = " << contador << endl;
        contador++;
    }
```

```
    cout << "\n---\n";

    for (contador = 0 ; contador <= 10 ; contador++) {
        cout << "Contador (for) = " << contador << endl;
    }
    return 0;
}
```

```
Overwriting aula01/ex24.cpp
```

In [73]:

```
ex='ex24'
!g++ aula01/"$ex".cpp -o aula01/$ex
!./aula01/$ex
```

```
Contador (while) = 0
Contador (while) = 1
Contador (while) = 2
Contador (while) = 3
Contador (while) = 4
Contador (while) = 5
Contador (while) = 6
Contador (while) = 7
Contador (while) = 8
Contador (while) = 9
Contador (while) = 10

---
Contador (for) = 0
Contador (for) = 1
Contador (for) = 2
Contador (for) = 3
Contador (for) = 4
Contador (for) = 5
Contador (for) = 6
Contador (for) = 7
Contador (for) = 8
Contador (for) = 9
Contador (for) = 10
```

switch (estrutura condicional)

In [74]:

```
%%file aula01/ex25.cpp
#include <iostream>
using namespace std;

int main() {
    int num;
    cout << "Digite um número: ";
    cin >> num;
    switch(num) {
        case 1:
            cout << "\nVocê digitou um!";
            break;
        case 2:
            cout << "\nVocê digitou dois!";
            break;
        case 3:
            cout << "\nVocê digitou três!";
            break;
        case 4:
            cout << "\nVocê digitou quatro!";
            break;
        case 5:
            cout << "\nVocê digitou cinco!";
            break;
```

```
        default:
            cout << "\nVocê digitou um número maior que cinco!";
            break;
    }
    cout << "\n";
    return 0;
}
```

```
Overwriting aula01/ex25.cpp
```

In [75]:

```
ex='ex25'
!g++ aula01/"$ex".cpp -o aula01/$ex
#!./aula01/$ex 11
```

Programa a ser executado até que o usuário digite 0!!!

In [76]:

```
%%file aula01/ex25b.cpp
#include <iostream>
using namespace std;

int main() {
    int num;
    do {
        cout << "Digite um número (0 para sair): ";
        cin >> num;
        switch(num) {
            case 1:
                cout << "\nVocê digitou um!";
                break;
            case 2:
                cout << "\nVocê digitou dois!";
                break;
            case 3:
                cout << "\nVocê digitou três!";
                break;
            case 4:
                cout << "\nVocê digitou quatro!";
                break;
            case 5:
                cout << "\nVocê digitou cinco!";
                break;
            default:
                cout << "\nVocê digitou um número maior que cinco!";
                break;
        }
        cout << "\n";
    } while (num != 0);
    return 0;
}
```

```
Overwriting aula01/ex25b.cpp
```

In [77]:

```
ex='ex25b'
!g++ aula01/"$ex".cpp -o aula01/$ex
#!./aula01/$ex 11
```

Vetores

In [78]:

```
%%file aula01/ex26.cpp
#include <iostream>
using namespace std;
```

```
int main() {
    int vetor[7];
    int i;

    // primeiro elemento
    vetor[0] = 2;
    // segundo elemento
    vetor[1] = 4;
    // último elemento
    vetor[6] = 4;

    // zerando todo o vetor
    for (i = 0 ; i < 7 ; i++)
        vetor[i] = 0;

    for (i = 0 ; i < 7 ; i++) {
        vetor[i] = i * 3;
    }

    for(i = 0; i < 7; i++) {
        cout << "\nValor do elemento vetor["
            << i
            << "] = "
            << vetor[i];
    }

    cout << "\n";

    // equivalente ao foreach de outras linguagens
    for (auto elemento: vetor) {
        cout << elemento << "\n";
    }

    return 0;
}
```

```
Overwriting aula01/ex26.cpp
```

In [79]:

```
ex='ex26'
!g++ aula01/"$ex".cpp -o aula01/$ex
!./aula01/$ex 11
```

```
Valor do elemento vetor[0] = 0
Valor do elemento vetor[1] = 3
Valor do elemento vetor[2] = 6
Valor do elemento vetor[3] = 9
Valor do elemento vetor[4] = 12
Valor do elemento vetor[5] = 15
Valor do elemento vetor[6] = 18
0
3
6
9
12
15
18
```

Matrizes

In [80]:

```
%%file aula01/ex27.cpp
#include <iostream>
using namespace std;
```

```cpp
int main() {
    int matriz2D[4][3] = { {2, 4 ,6},
                           {8, 10, 12},
                           {14, 16, 18},
                           {20, 22, 24}
                           };

    for(int i = 0; i < 4; i++) {
        for(int j = 0; j < 3; j++) {
            cout << matriz2D[i][j] << "\t";
        }
        cout << "\n";
    }
    return 0;
}
```

```
Overwriting aula01/ex27.cpp
```

In [81]:

```
ex='ex27'
!g++ aula01/"$ex".cpp -o aula01/$ex
!./aula01/$ex 11
```

```
2       4       6
8       10      12
14      16      18
20      22      24
```

Funções: parametros por referencia

In [82]:

```cpp
%%file aula01/ex28.cpp
#include <iostream>
using namespace std;

// passagem de parametro por valor
// o que acontece no parametro na função fica na função
void func1(int yp) {
  yp++;
  cout << "y em func1:\t" << yp << "\n";
}

// passagem de parametro por referencia
// o que acontece no parametro na função é retornado pelo parametro
void func2(int &yp) {
  yp++;
  cout << "y em func2:\t" << yp << "\n";
}

int main() {
    int y = 3;
    cout << "y em main: \t" << y << "\n";

    // passagem por valor: não altera y
    func1(y);
    cout << "y em main: \t" << y << "\n\n";

    // passagem por referencia: altera y
    func2(y);
    cout << "y em main: \t" << y << "\n";

    return 0;
}
```

```
Overwriting aula01/ex28.cpp
```

In [83]:
```
ex='ex28'
!g++ aula01/"$ex".cpp -o aula01/$ex
!./aula01/$ex
```

```
y em main:      3
y em func1:     4
y em main:      3

y em func2:     4
y em main:      4
```

Ponteiros

In [84]:
```
%%file aula01/ex29.cpp
#include <iostream>
using namespace std;

void func1(int y) {
  y++;
  cout << "valor de y em func1:\t" << y << "\n";
  cout << "endereço y: \t" << &y << "\n\n";
}

void func3(int *py) {

}

void func2(int *py) {
  (*py)++;
  cout << "valor de py em func2:\t" << *py << "\n";
  cout << "endereço py:\t" << py << "\n\n";
}

int main() {
    int y;
    int *py;

    y = 3;
    cout << "y = 3\n";
    cout << "valor de y em main: \t" << y << "\n";
    cout << "endereço y: \t" << &y << "\n\n";

    py = &y;
    cout << "py = &y;\n";
    cout << "valor de py em main: \t" << *py << "\n";
    cout << "endereço py: \t" << py << "\n\n";

    *py = 2;
    cout << "*py = 2;\n";
    cout << "valor de py em main: \t" << *py << "\n";
    cout << "endereço py: \t" << py << "\n\n";

    cout << "valor de y em main: \t" << y << "\n";
    cout << "endereço y: \t" << &y << "\n\n";

    cout << "func1(y);\n";
    func1(y);
    cout << "valor de y em main: \t" << y << "\n";
    cout << "endereço y: \t" << &y << "\n\n";

    cout << "func2(&y);\n";
    func2(&y);
    cout << "valor de y em main: \t" << y << "\n";
    cout << "endereço y: \t" << &y << "\n\n";
```

```
    cout << "func2(&y);\n";
    func2(&y);
    cout << "valor de y em main: \t" << y << "\n";
    cout << "endereço y: \t" << &y << "\n\n";
    cout << "valor de py em main: \t" << *py << "\n";
    cout << "endereço py: \t" << py << "\n\n";
}
```

```
Overwriting aula01/ex29.cpp
```

In [85]:

```
ex='ex29'
!g++ aula01/"$ex".cpp -o aula01/$ex
!./aula01/$ex
```

```
y = 3
valor de y em main:     3
endereço y:     0x7ffd1c6bbe84

py = &y;
valor de py em main:    3
endereço py:    0x7ffd1c6bbe84

*py = 2;
valor de py em main:    2
endereço py:    0x7ffd1c6bbe84

valor de y em main:     2
endereço y:     0x7ffd1c6bbe84

func1(y);
valor de y em func1:    3
endereço y:     0x7ffd1c6bbe6c

valor de y em main:     2
endereço y:     0x7ffd1c6bbe84

func2(&y);
valor de py em func2:   3
endereço py:    0x7ffd1c6bbe84

valor de y em main:     3
endereço y:     0x7ffd1c6bbe84

func2(&y);
valor de py em func2:   4
endereço py:    0x7ffd1c6bbe84

valor de y em main:     4
endereço y:     0x7ffd1c6bbe84

valor de py em main:    4
endereço py:    0x7ffd1c6bbe84
```

Struct

In [86]:

```
%%file aula01/ex30.cpp
#include <iostream>
using namespace std;

int main() {
    string nome;
    int idade;
    float salario;

    string nomeBob, nomePatrick;
```

```
    int idadeBob, idadePatrick;
    float salarioBob, salarioPatrick;

    struct Pessoa {
      string nome;
      int idade;
      float salario;
    };

    Pessoa bob;
    Pessoa patrick;

    bob.nome = "Bob Wilson";
    bob.idade = 33;
    bob.salario = 890.50;

    patrick.nome = "Patrick Estrela";

    cout << "Nome: " << bob.nome << "\n";
    cout << "Idade: " << bob.idade << "\n";
    cout << "Salario: " << bob.salario << "\n";
}
```

```
Overwriting aula01/ex30.cpp
```

In [87]:

```
ex='ex30'
!g++ aula01/"$ex".cpp -o aula01/$ex
!./aula01/$ex
```

```
Nome: Bob Wilson
Idade: 33
Salario: 890.5
```

In [88]:

```
%%file aula01/ex31.cpp
#include <iostream>
using namespace std;

int main() {
    struct Pessoa {
      string nome;
      int idade;
      float salario;
    };

    int j = 0;
    Pessoa pessoas[3];

    pessoas[0].nome = "Bob Wilson";
    pessoas[0].idade = 33;
    pessoas[0].salario = 890.50;
    j++;

    pessoas[1].nome = "Patrick Donald";
    pessoas[1].idade = 37;
    pessoas[1].salario = 900.70;
    j++;

    pessoas[2].nome = "Lula Mouse";
    pessoas[2].idade = 43;
    pessoas[2].salario = 1890.50;
    j++;

    for (int i = 0 ; i < j ; i++){
        cout << "Nome: " << pessoas[i].nome << "\n";
```

```
        cout << "Idade: " << pessoas[i].idade << "\n";
        cout << "Salario: " << pessoas[i].salario << "\n\n";
    }
}
```

```
Overwriting aula01/ex31.cpp
```

In [89]:

```
ex='ex31'
!g++ aula01/"$ex".cpp -o aula01/$ex
!./aula01/$ex
```

```
Nome: Bob Wilson
Idade: 33
Salario: 890.5

Nome: Patrick Donald
Idade: 37
Salario: 900.7

Nome: Lula Mouse
Idade: 43
Salario: 1890.5
```