Residência em Software

Estruturas de Controle de fluxo em C++

Professores:

Álvaro Coelho, Edgar Alexander, Esbel Valero e Hélder Almeida

INSTITUIÇÃO EXECUTORA

cepedi UESC

COORDENADORA

MCTI FUTURO Softex

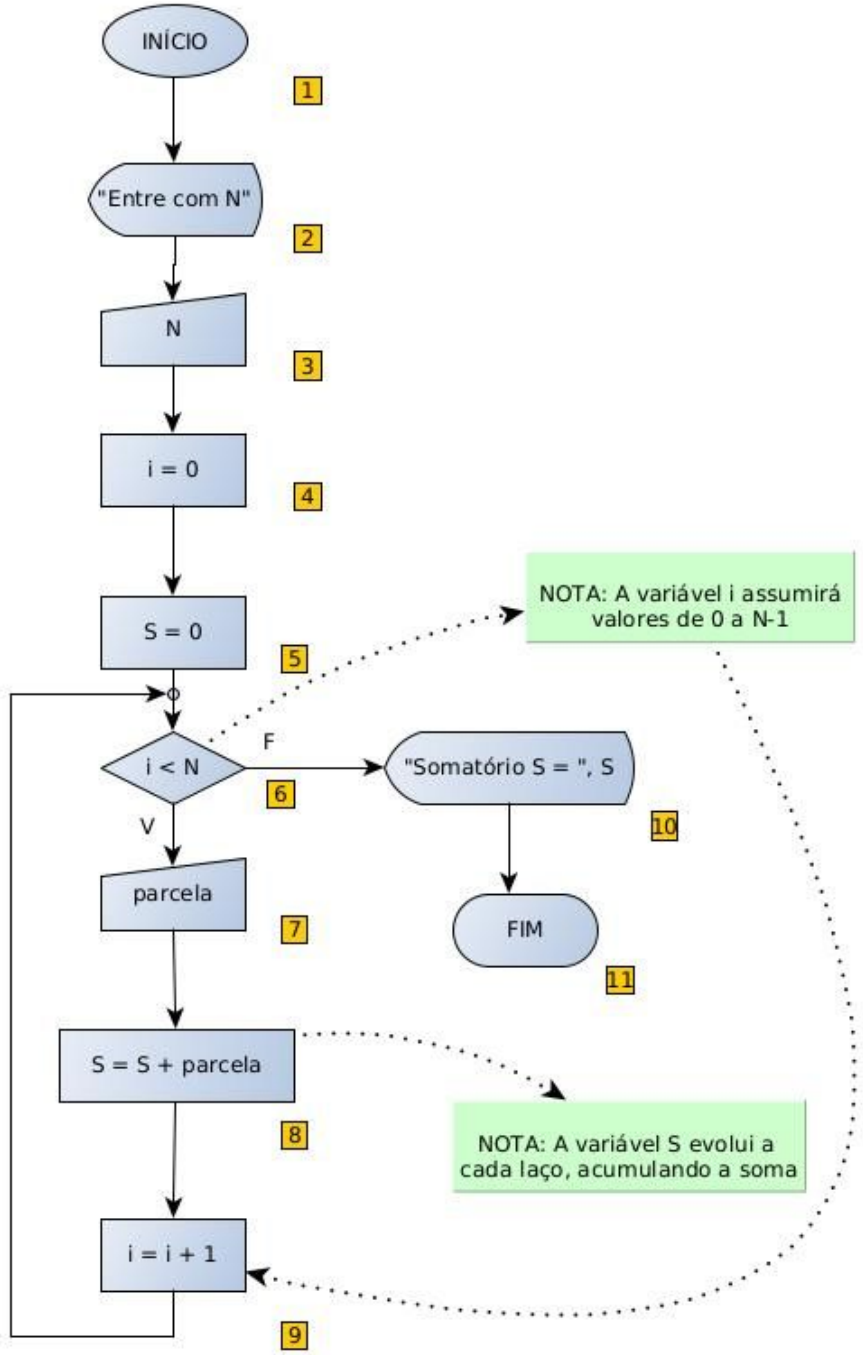
APOIO

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO

GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

Introdução

- O que são estruturas de controle de Fluxo em C++?
- Objetivo da aula: Aprender as estruturas de controle que permitem controlar o fluxo de execução de um programa em C++



Residência em Software

Recapitulando…

Relembrando os conceitos de tipos de dados básicos e operadores em C++

Residência em Software

Introdução

- Estruturas de controle são utilizadas para especificar a ordem em que as instruções devem ser executadas.

- Tipos de estruturas de controle:
  - estrutura sequencial,
  - estrutura condicional,
  - estrutura repetitiva.

Residência em Software

Estrutura sequencial

Como o próprio nome sugere, estrutura sequencial é um conjunto de instruções no qual cada uma será executada em sequência.

Obviamente, esta sequência obedece a uma lógica de programação.

A menos que seja especificado de outra forma, o computador executa automaticamente as instruções segundo a estrutura de sequência , isto é, uma após a outra, de cima para baixo e da esquerda para direita.

Residência em Software

Estruturas de Seleção

Permitem escolher blocos de instruções diferentes (caminhos diferentes durante a execução),

A linguagem C++ fornece três tipos de estruturas de seleção:

1. estrutura de seleção simples,

   ```
   if(condição){instruções}
   ```

2. estrutura de seleção dupla,

   ```
   if(condição){instruções}else{instruções}
   ```

3. estrutura de seleção múltipla.

   ```
   switch(expressão) { case valor: instruções; default: instruções; }
   ```

Residência em Software

Estrutura Condicional if

- Sintaxe básica: if (condição) { código }
- Verificar se uma pessoa é maior de idade

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

int main() {
    int idade;
    cout << "Digite a sua idade: ";
    cin >> idade;

    if (idade >= 18) {
        cout << "Voce e maior de idade." << endl;
    }

    cout << "Fim do programa." << endl;

    return 0;
}
```

Residência em Software

Estrutura Condicional if-else

- Sintaxe básica:

```
if (condição) {
    código
} else {
    código
}
```

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

int main() {
    int idade;
    cout << "Digite a sua idade: ";
    cin >> idade;

    if (idade >= 18) {
        cout << "Voce e maior de idade." << endl;
    } else {
        cout << "Voce e menor de idade." << endl;
    }

    return 0;
}
```

Residência em Software

Estrutura Condicional else if

- Sintaxe básica:

```
if (condição) {
    código
} else if (condição) {
    código
} else {
    código
}
```

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

int main() {
    int numero;

    cout << "Digite um numero: ";
    cin >> numero;

    if (numero < 0) {
        cout << "O numero " << numero << " e negativo." << endl;
    } else if (numero > 0) {
        cout << "O numero " << numero << " e positivo." << endl;
    } else {
        cout << "O numero e zero." << endl;
    }

    return 0;
}
```

Residência em Software

Exemplo prático

Classificar notas em conceitos (A, B, C, D, F)

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

int main() {
    int nota;

    cout << "Digite a nota (0-10): ";
    cin >> nota;

    if (nota >= 0 && nota <= 10) {
        if (nota >= 9) {
            cout << "Conceito A" << endl;
        } else if (nota >= 7) {
            cout << "Conceito B" << endl;
        } else if (nota >= 5) {
            cout << "Conceito C" << endl;
        } else if (nota >= 3) {
            cout << "Conceito D" << endl;
        } else {
            cout << "Conceito F" << endl;
        }
    } else {
        cout << "Nota invalida. Digite um valor entre 0 e 10." << endl;
    }

    return 0;
}
```

Residência em Software

Estrutura condicional Switch

O switch testa sucessivamente o valor de uma expressão contra uma lista de constantes inteiras, quando o valor coincide, os comandos associados àquela constante são executados.

Embora case seja uma palavra reservada da linguagem ela não pode se utilizada fora da estrutura switch .

O switch só pode testar igualdade.

Duas constantes case no mesmo switch não podem ter valores idênticos.

Quando constantes caracteres são usadas no comando switch elas são, automaticamente, convertidas em inteiros.

Residência em Software

Estrutura condicional Switch

O comando break, é um comando de desvio, quando encontrado a execução continua na primeira linha de código após o switch .

O comando default é executado se nenhuma coincidência for detectada.

O default é opcional, se não estiver presente nenhuma ação será realizada se todos os testes falharem.

Tecnicamente, os comandos break são opcionais, se o break for omitido, a execução continua pelos próximos comandos até que um break ou o fim do switch seja encontrado. A omissão do break pode levar a erros de lógica na execução.

```
int main() {
    char operacao;
    double num1, num2;

    cout << "Digite a operacao (+, -, *, /): ";
    cin >> operacao;

    cout << "Digite dois numeros: ";
    cin >> num1 >> num2;

    switch (operacao) {
        case '+':
            cout << "Resultado: " << num1 + num2 << endl;
            break;
        case '-':
            cout << "Resultado: " << num1 - num2 << endl;
            break;
        case '*':
            cout << "Resultado: " << num1 * num2 << endl;
            break;
        case '/':
            if (num2 != 0) {
                cout << "Resultado: " << num1 / num2 << endl;
            } else {
                cout << "Erro: Divisao por zero." << endl;
            }
            break;
        default:
            cout << "Operacao invalida." << endl;
    }

    return 0;
}
```

Residência em Software

Estrutura de Repetição while

Permitem ao programador especificar que uma ação deve ser repetida enquanto uma determinada condição for verdadeira.

Sintaxe básica:

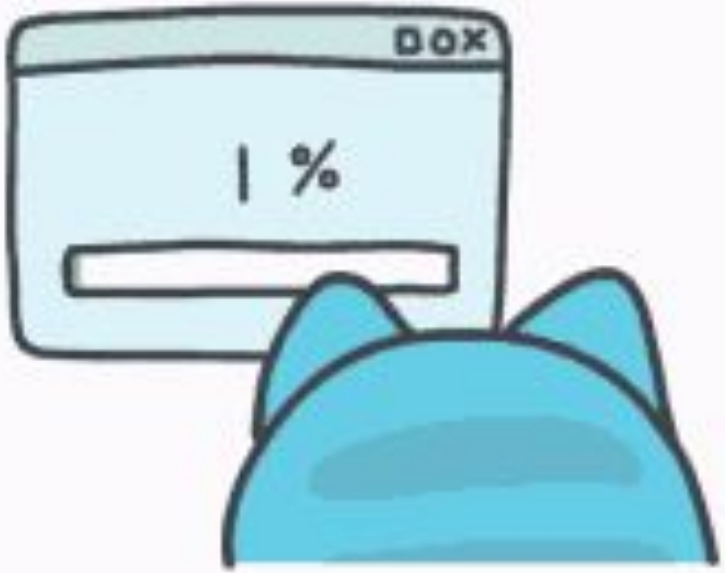
```
while (condição) {
   código
}
```

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

int main() {
    int contador = 1;

    while (contador <= 5) {
        cout << "Contador: " << contador << endl;
        contador++;
    }

    return 0;
}
```

Exemplo prático - Contagem regressiva

Considere um programa para imprimir a primeira potência de 2 maior que 500. O código abaixo está correto?

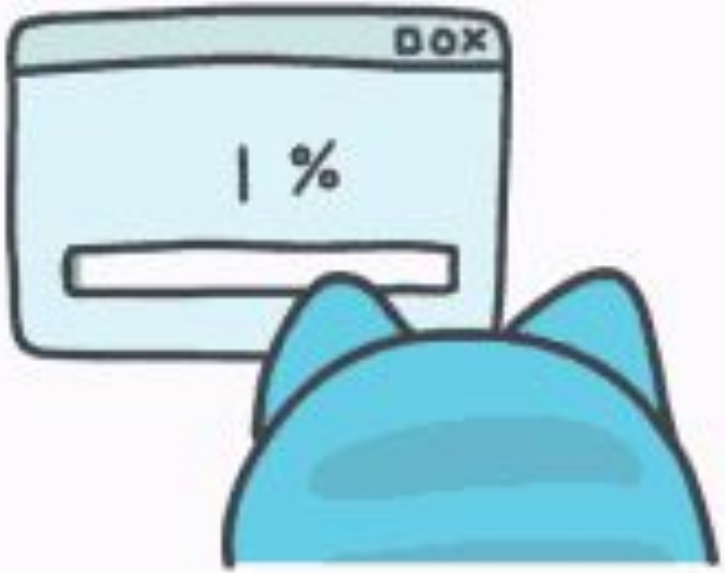
```
int main(){
  int prod;
  while (prod <= 500){
    prod = prod * 2;
  }
  printf("%d",prod);
}
```



Exemplo prático - Contagem regressiva

Considere um programa para imprimir a primeira potência de 2 maior que 500. O código abaixo está correto?

```
int main(){
  int prod;
  while (prod <= 500){
    prod = prod * 2;
  }
  printf("%d",prod);
}
```



Onde está a inicialização da variável????

Cuidado com repetições infinitas!!!!!

Residência em Software

Estrutura de Repetição do-while

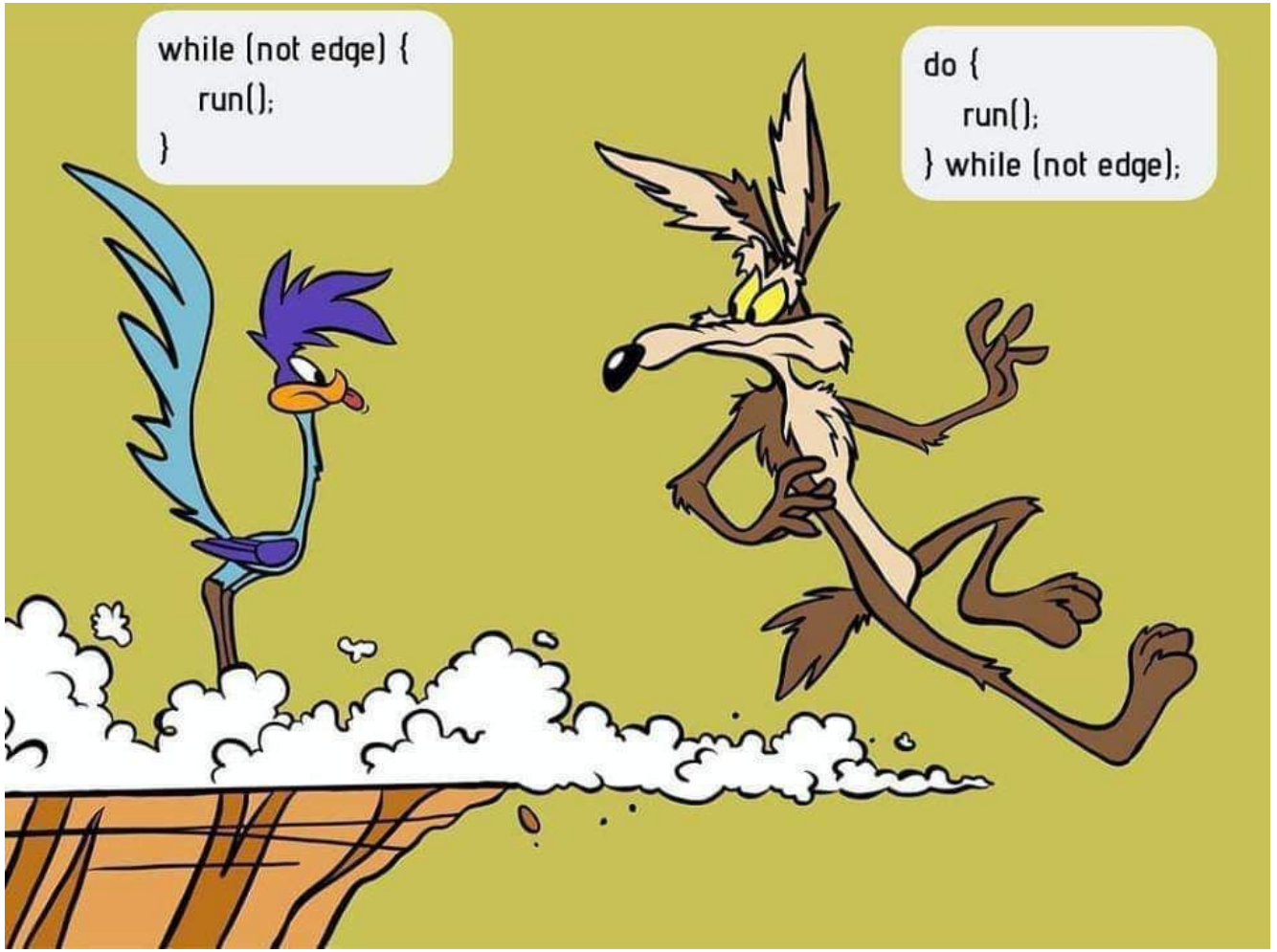
Sintaxe básica:

```
do {
   código
} while (condição)
```

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

int main() {
    int contador = 1;

    do {
        cout << "Contador: " << contador << endl;
        contador++;
    } while (contador <= 5);

    return 0;
}
```

Residência em Software

While vs Do - While



Residência em Software

Estrutura de Repetição do-while

Sintaxe básica:

```
do {
   código
} while (condição)
```

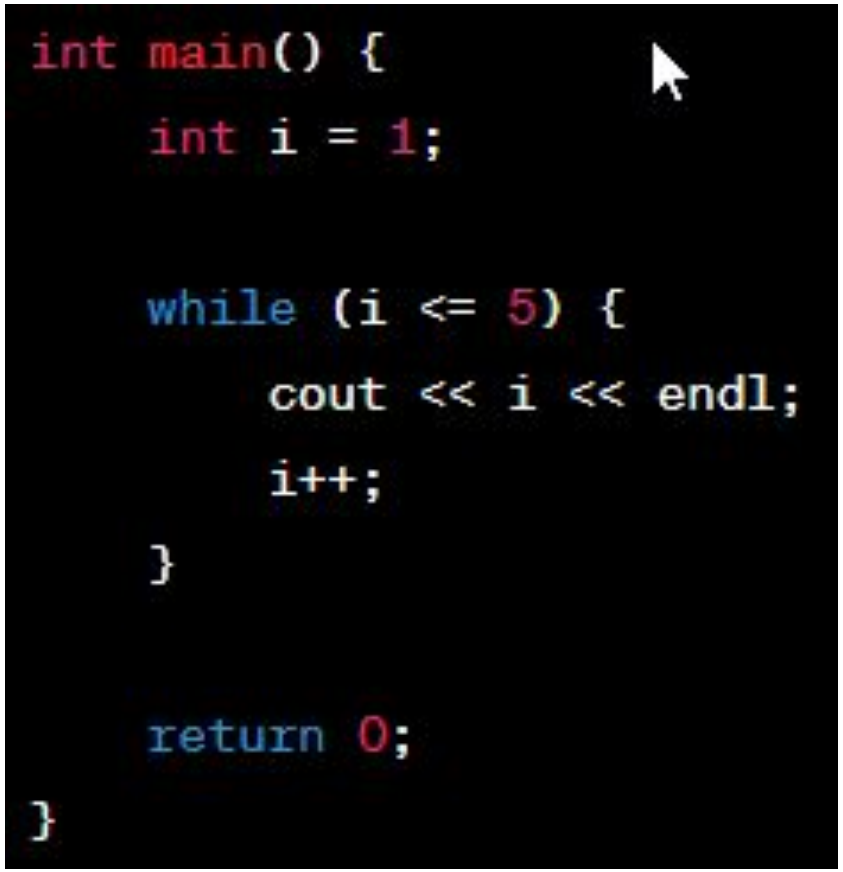
```cpp
#include <iostream>
using namespace std;

int main() {
    int contador = 1;

    do {
        cout << "Contador: " << contador << endl;
        contador++;
    } while (contador <= 5);

    return 0;
}
```

Residência em Software

Conceito de contador

Repetição controlada por contador

Estruturas de controle em programação que permitem que um bloco de código seja executado várias vezes com base em um contador ou variável de controle. Essa variável de controle é incrementada ou decrementada a cada iteração do loop, e o loop continua enquanto a condição relacionada ao contador é verdadeira.



```
int main() {
    int i = 1;

    while (i <= 5) {
        cout << i << endl;
        i++;
    }

    return 0;
}
```

Residência em Software

Praticando repetição por contador…

Elabore um algoritmo para resolver o problema:

Uma turma de dez alunos fez um teste, leia as notas (inteiros de 0 a 100) da turma e determine a média da turma no teste.

Residência em Software

Conceito de sentinela

Repetição controlada por sentinela

Estruturas de controle em programação que permitem que um bloco de código seja executado repetidamente enquanto uma condição específica for verdadeira. O "sentinela" aqui é uma condição que atua como um sinal para interromper a repetição, comumente chamado de “*flag*”

```
int main() {
    int numero;

    cout << "Digite numeros positivos (ou negativos para sair):" << endl;

    cin >> numero;

    while (numero >= 0) {
        cout << "Voce digitou: " << numero << endl;
        cin >> numero;
    }

    cout << "Saindo do programa..." << endl;

    return 0;
}
```

Residência em Software

Conceito de acumulador

O conceito de acumuladores em laços de repetição está relacionado à prática de somar ou acumular valores ao longo de várias iterações de um loop. Um acumulador é uma variável que é usada para armazenar a soma (ou acumulação) de valores ao longo do tempo enquanto o loop é executado.

Os acumuladores são uma ferramenta poderosa para calcular resultados complexos ao longo de interações repetidas em um loop. Eles ajudam a simplificar o processo de cálculo e a tornar o código mais legível e eficiente.

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

int main() {
    int total = 0;
    int numero;

    cout << "Digite uma sequencia de numeros (insira 0 para sair):" << endl;

    do {
        cin >> numero;
        total += numero; // Acumulando os valores
    } while (numero != 0);

    cout << "A soma total dos numeros inseridos e: " << total << endl;

    return 0;
}
```

Residência em Software

Praticando repetição por sentinela e acumulador…

Elabore um algoritmo para resolver o problema:

Calcular a média de uma turma num teste, o programa deve processar um número arbitrário de alunos cada vez que for executado. As notas serão valores inteiros entre 0 e 100. Em cada iteração, o algoritmo deve perguntar se deseja prosseguir.

Residência em Software

Teste!

O que faz esse algoritmo?!

```
int main(){
  int num1, num2, res=0, i=0, s, s1=1, s2=1;
  cout << "Digite o primeiro numero: ";
  cin >> num1;
  cout << "Digite o primeiro numero: ";
  cin >> num2;
  if (num1 < 0){
    num1 = -num1;
    s1 = -1;
  }
  if (num2 < 0){
    num2 = -num2;
    s2 = -1;
  }
  s = s1 * s2;
  while(i < num2){
    res += num1;
    i++;
  }
  cout << "Resultado " << res*s << endl;
  return 0;
}
```

Residência em Software

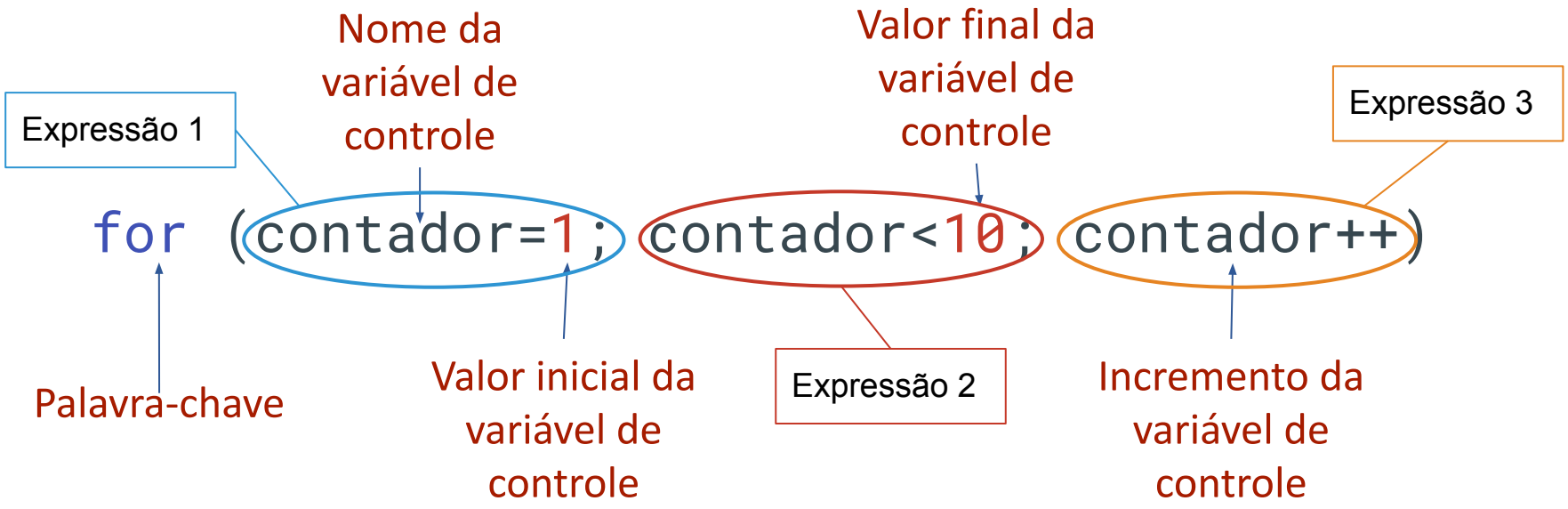
Estrutura de Repetição for

- Vimos que na repetição controlada por contador é necessário:
  1. A **inicialização** de uma variável de controle,
  2. A **condição** que testa o valor final da variável de controle.
  3. O **incremento** (ou **decremento**) pelo qual a variável de controle é modificada cada vez que o corpo do laço é realizado.

```
int contador = 1; //inicialização
while (contador <= 10) { //condição
   cout << contador << " ";
   contador++; //incremento
}
```

Estrutura de Repetição for

A estrutura de repetição **for** manipula automaticamente todos os detalhes da repetição controlado por contador.




```
int contador;
for(contador=1; contador <= 10; contador++){
   cout << contador << " ";
}
```

Residência em Software

Estrutura de Repetição for

1. Frequentemente **Expressão 1**, **Expressão 2** e **Expressão 3** são expressões múltiplas separadas por vírgulas, onde a lista de expressões é avaliada da esquerda para a direita.
2. As três expressões da estrutura **for** são opcionais e podem ser omitidas.
3. Se a **Expressão 2** for omitida, a linguagem C++ presume que a condição é verdadeira e cria um loop infinito.
4. A **Expressão 1** pode ser omitida se a variável de controle for inicializada em outro lugar do programa.
5. A **Expressão 3** pode ser omitida se o incremento é calculado no corpo da estrutura for ou se nenhum incremento se faz necessário.

Residência em Software

Estrutura For

1. A inicialização, condição e continuação do loop podem conter operações aritméticas.

```
x = 2; y = 10
for(j=x; j<=4*x*y; j+=y/x)

for(j=2; j<=80; j+=5)
```

2. O “incremento” pode ser negativo, neste caso temos decremento ou contagem regressiva.

```
for(i=10; i>=0; i--)
  cout << i << endl;
```

Residência em Software

Estrutura For

3. Se a condição de continuação for inicialmente falsa, o corpo do loop nunca é executado, a execução continua na próxima instrução após o loop.

```
for(i=k; i<k; i++)
  cout << i << endl;
```

4. Laço infinito.

```
for(;;){
  cin >> ch;
  if (ch=='A') break;
}
```

Residência em Software

Estrutura For

5. Laço for com condição composta

```
flag = 1;
  for(i=0; i<10 && flag; i++){
  cout << “Digite um numero positivo:”;
  cin >> num;
  if (num<0) flag = 0;
}
```

6. Laço for como retardo de tempo

```
for(i=0; i<100000; i++);
```

Residência em Software

Estrutura For

7. Laço for sem corpo

```
for(soma=0, num=2; num<=100; soma+=num, num+=2);
```

```
soma=0;
for(num=2; num<=100; num=num+2)
  soma+=num;
```

8. Laço for utilizando caracteres

```
for(char letra='A'; letra<='Z'; letra++)
  cout << int(letra) << letra;
```

Residência em Software

Exercícios

1. Imprimir a tabuada de um número fornecido pelo usuário
2. Crie um programa para processar as notas de um exame cujos conceitos são: A, B, C, D ou F, e, ao final, informe a quantidade de alunos que obtiveram cada nota; após o cadastro de cada nota o programa deve perguntar ao usuário se deseja continuar.
3. Desenvolver um jogo em que o usuário deve adivinhar um número entre 1 e 1000 e o computador informa se acertou, se é menor ou maior que o número sorteado. O jogador terá 5 chances para adivinhar.

Residência em Software

Exercício

Uma pessoa investe 1000 reais em uma conta de poupança que rende juros do 5 por cento. Admitindo que todos os juros são deixados em depósito na conta, calcule e imprima a quantia na conta ao final de cada ano, ao longo dos anos. Use a fórmula de juros compostos:

$$a = p(1 + r)^n$$

p, quantia investida originalmente

r, taxa de juros anual,

n, número de anos,

a, quantia existente no final

Residência em Software

Conclusão

- Recapitulação dos principais conceitos abordados na aula
- Destaque para a importância das estruturas de controle no desenvolvimento de programas em C++