


<div><p><b>Inatel</b> <i>Instituto Nacional de Telecomunicações</i></p></div>	Relatório 1	Data:    /    /	
	Disciplinas: E209		
	Profs: João Magalhães e Yvo Chiaradia Monitores: Thalita Domingos, Diego Coutinho, Pedro Fraga, Thayana Lucero e Ewel Fernandes		
Conteúdo: Revisão da Linguagem C			
Tema: Estruturas de decisão e repetição, loop infinito e funções			
Nome:		Matricula:	Curso:

### Objetivos:

- Revisar os conceitos básicos da linguagem C;
- Execução de exemplos de programas através de simuladores.

### *Parte Teórica*

Nesse relatório serão revisados os principais conceitos de programação com linguagem C, que serão utilizados no decorrer do semestre.

### Estruturas de Decisão:

Na programação, nos deparamos com diversos tipos de problemas em que precisamos estabelecer desvios(decisões) nas execuções dos comandos para conseguirmos resolvê-los.

- Estrutura de Decisão **SE – SENÃO (if – else)**: usada quando se tem trechos do código que só podem ser executadas **SE** uma condição for verdadeira.

```
//simples
if(condição) //se a condição for verdadeira
    comando; //executa aqui
```

```
//composto
if(condição) //se a condição for verdadeira
    comando; //executa aqui
else //senão
    comando; //executa aqui
```

- **Switch-case**: estrutura alternativa ao if – else. Usado quando se quer escolher uma dentre várias opções. Contém os comandos **case** (para a codificação de cada caso) e **break** (para interromper a estrutura).

```

switch (variavel){           // "Testamos" a variavel desejada

    case constante1:        // caso seja a constante 1
        comandos;           // executa aqui até encontrar o break
        break;              // sai da estrutura switch

    case constante2:        // caso seja a constante 2
        comandos;           // executa aqui até encontrar o break
        break;              // sai da estrutura switch

    case constante3:        // caso seja a constante 3
        comandos;           // executa aqui até encontrar o break
        break;              // sai da estrutura switch

    default:                // caso não seja nenhuma das constantes acima
        comandos;           // executa aqui até encontrar o break
        break;              // sai da estrutura switch
}

```

## Estruturas de Repetição:

As estruturas de Repetição não só nos permitem executar bloco de comandos sob determinadas condições, mas também de repetir o mesmo bloco quantas vezes for necessário. Essas estruturas são úteis para repetir uma série de comandos semelhantes ou simplesmente repetir um mesmo processamento até que uma certa condição seja satisfeita.

- Estrutura de Repetição **Enquanto (While)**: usada quando se tem trechos de código que podem ser executados repetidas vezes, **ENQUANTO** uma condição é verdadeira.

```

while(condição){           // Enquanto a condição for verdadeira

    comandos;               // Executa os comandos dentro das {}
    comando1;
    comando2;

    incremento;             // incremento de alguma variável
                             // caso necessário
}

```

- Estrutura de Repetição **Para (for)**: estrutura alternativa ao while. Usada quando se tem um trecho de código que deve ser executado uma quantidade pré-determinada de vezes. Possui três itens configuráveis em sua estrutura: **condição inicial**, **condição de parada e incremento**. Usamos uma variável contadora do tipo inteira e determinamos uma condição inicial para ela. **PARA** toda vez que a condição de parada for verdadeira, o trecho de código dentro do **for** será executado e, no fim, a variável será incrementada.

```
for(inicializacao; condição de parada; incremento) {
    comandos;
}
```

No exemplo abaixo, quantas vezes o trecho dentro do for foi executado?

```
int x;
for(x=1; x<=5; x++) {
    cout << x << endl;
}
```

Diagrama de anotações no código acima:

- inicialização** (seta roxa) aponta para `x=1`.
- teste** (seta laranja) aponta para `x<=5`.
- incremento** (seta azul) aponta para `x++`.

## Loops infinitos:

É possível criar loops infinitos utilizando for e while. Esse tipo de loop será muito utilizado nas aulas de E209. Mas como fazer isso?

- **Com for:** se você não estabelecer nenhuma condição inicial ou de parada, seu código rodará infinitamente!

```
for(;;){
    comandos;
}
```

- **Com while:** quando você estabelece que a condição da sua estrutura é sempre verdadeira, seu código também rodará infinitamente!

```
while(1){
    comandos;
}
```

## Funções:

Funções são usadas para criar pequenos pedaços de códigos separados do programa principal. Em C e C++, `int main` é a nossa função principal. Exceto a função `main`, todas as outras funções são secundárias, o que significa que elas podem existir ou não.

Existem alguns tipos de funções: **Passagem de Parâmetro por Valor, Passagem de Parâmetro por Referência, Sem Retorno, Sem Parâmetro.**

- **Passagem de Parâmetro por Cópia:** também chamada de passagem por valor, a função recebe apenas uma cópia dos valores passados a ela. Se o valor da variável for alterado, a mudança afeta apenas a função.

```

int soma(int a, int b){
    int resultado;

    resultado = a + b;

    return resultado;
}

int main()
{
    int n1, n2, total;

    cout << "Entre com dois numeros inteiros" << endl;
    cin >> n1 >> n2;

    total = soma(n1,n2);

    cout << "A soma eh: " << total << endl;

    return 0;
}

```

- **Função sem retorno:** São funções do tipo **void**, em que não há retorno de valores. A saída do programa está, na maioria das vezes, na própria função.

```

void soma(int a, int b){
    cout << "A somam eh: " << a+b << endl;
}

int main()
{
    int n1, n2;

    cout << "Entre com dois numeros" << endl;
    cin>> n1 >> n2;

    soma(n1, n2);

    return 0;
}

```

- **Função Sem Parâmetro:** Não há valores passados para a função.

```
int multiplica(){
    int a = 3;
    int b = 2;

    return a*b;
}

int main()
{
    cout << "3 x 2 = " << multiplica() << endl;

    return 0;
}
```

## Parte Prática

**Questão 1)** Em uma banca de frutas o preço das laranjas pode assumir 2 valores distintos, R\$0,80 cada se forem adquiridas menos de uma dúzia, e R\$0,65 se forem adquiridas pelo menos doze. Utilizando a estrutura de decisão if-else, elabore um programa que leia o número de laranjas compradas e imprima o valor total da compra.

**OBS.: Limitador de casas decimais em C: #include <iomanip>**  
**cout << fixed << setprecision(numero\_de\_casas\_decimais);**

Ex.:

**Entrada:**12

**Saída:**

Preço da unidade: R\$0.65

Preço total: R\$7.80

**Entrada:** 6

**Saída:**

Preço da unidade: R\$0.80

Preço total: R\$4.80

**Questão 2)** Desenvolva um código que leia um valor numérico entre 0 e 50, inclusive, e imprima o resto da divisão desse número por todos os seus antecessores maiores que 0.

**Atenção: Faça a crítica de dados para que o programa só aceite os valores citados. Lembre-se de que não existe divisão por 0!**

Ex.: **Entrada:** 30

**Saída:**

Resto da divisao de 30 por 29: 1

Resto da divisao de 30 por 28: 2

Resto da divisao de 30 por 27: 3

```
Resto da divisao de 30 por 26: 4
Resto da divisao de 30 por 25: 5
Resto da divisao de 30 por 24: 6
Resto da divisao de 30 por 23: 7
Resto da divisao de 30 por 22: 8
Resto da divisao de 30 por 21: 9
Resto da divisao de 30 por 20: 10
Resto da divisao de 30 por 19: 11
Resto da divisao de 30 por 18: 12
Resto da divisao de 30 por 17: 13
Resto da divisao de 30 por 16: 14
Resto da divisao de 30 por 15: 0
Resto da divisao de 30 por 14: 2
Resto da divisao de 30 por 13: 4
Resto da divisao de 30 por 12: 6
Resto da divisao de 30 por 11: 8
Resto da divisao de 30 por 10: 0
Resto da divisao de 30 por 9: 3
Resto da divisao de 30 por 8: 6
Resto da divisao de 30 por 7: 2
Resto da divisao de 30 por 6: 0
Resto da divisao de 30 por 5: 0
Resto da divisao de 30 por 4: 2
Resto da divisao de 30 por 3: 0
Resto da divisao de 30 por 2: 0
Resto da divisao de 30 por 1: 0
```

**Questão 3)** Faça um código que recebe um valor *n*, referente a quantidade de testes que serão feitos. Logo em seguida, monte uma função que recebe 5 valores inteiros e mostra quantos valores digitados foram pares, quantos valores digitados foram ímpares, quantos valores digitados foram positivos e quantos valores digitados foram negativos e mostre na tela.

Ex.:

**Entrada:**

2

-2 1 33 14 111

-13 72 90 -1 0

**Saída:**

Quantidade de numeros pares: 2

Quantidade de numeros impares: 3

Quantidade de numeros positivos: 4

Quantidade de numeros negativos: 1

Quantidade de numeros pares: 3

Quantidade de numeros impares: 2

Quantidade de numeros positivos: 3

Quantidade de numeros negativos: 2