

---

# Estruturas de repetição

## Controle de Desvios

Prof. Karl Apaza Agüero

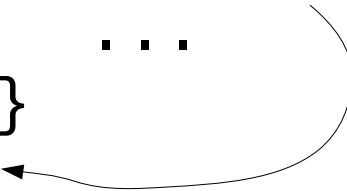
---

# Controle de desvios

---

- ▶ **break** (sair ou quebrar):
  - ▶ Finaliza a execução de um *loop* incondicionalmente
    - Termina as iterações da estrutura de *loop* sem verificar se a condição ainda é satisfeita ou não

```
for(;;)
{
    ...
    break;
    ...
}
```



# Controle de desvios

---

## ► break (sair ou quebrar):

### ► Exemplos:

```
#include <stdio.h>
int main(){
    for(;;){
        printf("Digite 1 para
               finalizar o for\n");
        int n;
        scanf("%d",&n);
        if(n==1){ break;}
    }
    return 0;
}
```

```
#include <stdio.h>
int main() {
    int num;
    for(int i=0; i<100; i++){
        scanf("%d", &num);
        if (num<0){ break;}
        printf("%d\n", num);
    }
    return 0;
}
```

## PROBLEMA: USAR BREAK

Um Dadinho é um doce muito popular no Brasil, por ser gostoso e barato. É tão barato que Rodovalho, sempre que toma café com seus amigos, compra vários Dadinhos para serem comidos enquanto o café é bebido.

Há **N** pessoas no grupo de amigos de Rodovalho, e ele quer comprar **D** dadinhos para cada pessoa. Considerando que cada dadinho custa **C** centavos, quanto Rodovalho irá gastar?

## ENTRADA

A entrada consiste em vários casos de teste. Cada caso contém uma única linha contendo três valores, **N**, **C** e **D** ( $1 \leq \mathbf{N}, \mathbf{C}, \mathbf{D} \leq 100$ ). A entrada termina com **N=C=D=0**.

## SAIDA

Para cada caso de teste, imprima uma única linha contendo um único inteiro, indicando quanto Rodovalho irá gastar, em centavos.

## EXEMPLOS

Entrada	Saída
3 2 15 1 1 1 0 0 0	90 1

---

```
#include <iostream>
using namespace std;
int main(){
    for(;;){
        int N,C,D;
        cin>>N>>C>>D;
        if(N==0 && C==0 && D==0){break;}
        cout<<N*C*D<<endl;
    }
    return 0;
}
```

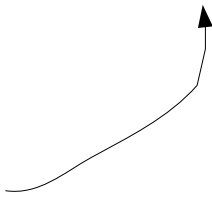
---

# Controle de desvios

---

- ▶ `continue` (voltar):
  - ▶ Executa a próxima iteração sem executar o restante do laço

```
for(i=0; i<10; i++)  
{  
    ...  
    continue;  
    ...  
}
```



# Controle de desvios

---

## ► continue (voltar):

### ► Exemplos:

```
#include <stdio.h>
#define MAX 10
int main(){
    int c;
    for(c=1; c<=MAX; c++){
        printf("\n%d\n",c);
        if(c%2==0){ continue;}
        printf("Impar\n");
    }
    return 0;
}
```

```
#include <stdio.h>
int main() {
    int num;
    for(int i=0; i<100; i++){
        scanf("%d", &num);
        if (num<0){continue;}
        printf("%d\n", num);
    }
    return 0;
}
```

# Controle de desvios

---

## ► Problema: usar continue

- Fazer um programa para gerar a série de Fibonacci:

1 1 2 3 5 8 13 21 34 55 89

até o n-ésimo elemento informado pelo usuário ( $n > 0$ ).



```
#include <iostream>
using namespace std;
int main(){
    int n;
    cin>>n;
    int a=1, b=1;
    for(int i=0; i<n; i++){
        if(i<2){
            cout<<"1 ";
            continue;
        }
        int c=a+b;
        cout<<c<<" ";
        a=b;
        b=c;
    }
    cout<<endl;
    return 0;
}
```

# PROBLEMA

Os alunos do último ano resolveram organizar uma quermesse para arrecadar fundos para a festa de formatura. A festa prometia ser um sucesso, pois o pai de um dos formandos, Teófilo, dono de uma loja de informática, decidiu doar um computador para ser sorteado entre os que comparecessem. Os alunos prepararam barracas de quentão, pipoca, doces, ensaiaram a quadrilha e colocaram à venda ingressos numerados sequencialmente a partir de 1. O número do ingresso serviria para o sorteio do computador.

Depois da entrada de todos os participantes, Teófilo começou a trabalhar no computador para preparar o sorteio. Verificando a lista de presentes, notou uma característica notável: havia apenas um caso, em toda a lista, em que o participante que possuía o ingresso numerado com  $i$ , havia sido a  $i$ -ésima pessoa a entrar no ginásio.

**Tarefa:** Conhecendo a lista de participantes, por ordem de chegada, sua tarefa é determinar o número do ingresso premiado, sabendo que o ganhador é o único participante que tem o número do ingresso igual à sua posição de entrada na festa.

**Entrada:** A entrada é composta de vários conjuntos de teste. A primeira linha de um conjunto de teste contém um número inteiro positivo  $N$  que indica o número de participantes da festa. A linha seguinte contém a sequência, em ordem de entrada, dos  $N$  ingressos das pessoas que participaram da festa. O final da entrada é indicado quando  $N = 0$ . Para cada conjunto de teste da entrada haverá um único ganhador.

**Saída:** Para cada conjunto de teste da entrada seu programa deve produzir três linhas. A primeira linha identifica o conjunto de teste, no formato "Teste  $n$ ", onde  $n$  é numerado a partir de 1. A segunda linha deve conter o número do ingresso do ganhador, conforme determinado pelo seu programa. A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída deve ser seguida rigorosamente.

## Exemplo

### Entrada:

```
4
4 5 3 1
10
9 8 7 6 1 4 3 2 12 10
0
```

### Saída:

```
Teste 1
3

Teste 2
10
```

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

int main() {

    for(int t=1; ;t++){
        int n;
        cin>>n;
        if(n==0){break;}

        int ni, nig;
        for(int i=1; i<=n; i++){
            cin>>ni;
            if(ni==i){nig=ni;}
        }

        cout<<"Teste "<<t<<endl;
        cout<<nig<<endl<<endl;
    }

    return 0;
}
```