# Introdução ao C++

Curso OBI 2023
Avançados - Alan Hahn Pereira

# Por que C++?

# Performance!

# O mesmo código em duas linguagens

C++

```cpp
#include <stdio.h>
#define MOD 100000009

long long fib(int a){
    if (a <= 1) return 1;
    return (fib(a-1) + fib(a-2))%MOD;
}

int main(){
    printf("%lld\n", fib(35));
}
```

Tempo de Execução:
79 ms

Python

```python
MOD = 1000000009

def fib (a):
    if (a<=1):
        return 1
    return (fib(a-1)+fib(a-2))%MOD

print(fib(35))
```

Tempo de Execução:
2783 ms

# Motivos por escolher C++ para OBI

- C++ é uma linguagem compilada
- Linguagem de baixo nível
- Prós
  - Rápido para rodar depois de compilado
  - Alto nível de controle
- Contras
  - Tem que compilar
  - As vezes é necessário escrever mais para fazer a mesma coisa

- Python é uma linguagem interpretada
- Linguagem de alto nível
- Prós
  - Não é necessário compilar
  - Rápido para codar
- Contras
  - Lento para rodar
  - Coisas ficam ocultas do programador

# Tipagem em C++

# Tipagem em C++

O bit é unidade fundamental de dispositivos digitais

O byte é composto de 8 bits e é a unidade básica de todos os computadores

Tipos fundamentais do C++:

- Char: 1 byte com 1 bit de sinal (-128 a 127)
  - Frequentemente usado para representar caracteres
- Short int: 2 bytes com 1 bit de sinal (-32768 a 32767)
- Int: 4 bytes com 1 bit de sinal (-2147483648 a 2147483647)
- Long long: 8 bytes com 1 bit de sinal ( -9e18 a 9e18)

# Tipagem em C++

Ponto Flutuante

- Tem precisão variável
- Representação em notação científica em binário: 1,... * 2 ^M
- Perde precisão absoluta quanto maior o valor da variável
- Pode representar números pequenos com bastante precisão ou números grandes sem muita precisão
- Float: 32 bits: 1 sinal, 8 expoente, 23 mantissa
- Double: 64 bits: 1 sinal, 11 expoente, 52 mantissa

# Básicos do C++

# Entrada e saída

- scanf(“%d”, &x);
  - Lê um inteiro para variável x
- scanf(“%d %d”, &x, &y);
  - Lê dois inteiros para variáveis x e y
- Existem outras opções para o scanf

- cin >> x;
  - Lê um inteiro para variável x
- cin >> x >> y
  - Lê dois inteiros para variáveis x e y
- Não é necessário explicitar o tipo do valor a ser lido

- printf(“%d\n”, x);
  - Imprime a variável x
- printf(“O valor de x e %d\n”, x);
  - Imprime a variável x com texto adicional

- cout << x << endl;
  - Imprime a variável x
- cout << “O valor de x e” << x << endl;
  - Imprime a variável x com texto adicional

# Declaração de variáveis

- int a;
  - Declara a variável ‘a’ do tipo int
- int a = 0;
  - Declara a variável ‘a’ do tipo int com valor inicial 0
- int a, b;
  - Declara a variável ‘a’ e ‘b’ do tipo int
- int a = 0, b= 5;
  - Declara a variável ‘a’ e ‘b’ do tipo int com valor inicial 0 e 5

# Operadores matemáticos

- a+b
  - Soma
- a-b
  - Subtração
- a*b
  - Multiplicação
- a/b
  - Divisão
- a%b
  - Módulo (Resto da divisão inteira de A por B)

# Comparadores

- a == b
  - Confere se A é igual B
- a != b
  - Diferente
- a < b
  - Menor
- a > b
  - Maior
- a <= b
  - Menor ou igual
- a >= b
  - Maior ou Igual

# Operadores Lógicos

- a && b
  - “E” Lógico
- a || b
  - “Ou” Lógico
- !a
  - “Não” Lógico
- (c > d) != (e > f)
  - “Ou exclusivo” (XOR) Lógico

# Condicionais

```
if(condição){
        ações
}
else if(condição2){
}
else{
}
```

```
if(condição) ação;
else açãoAlternativa;
```

# Loops

while(condição)

- repete o bloco de dentro enquanto a condição for verdadeira no começo de cada repetição
- Se a condição for falsa em algum momento no interior da execução mas quando chegar no final do loop e for verdadeira novamente não sai do while)

for(int i = 0; i < n; i++)

- executa o primeiro bloco antes do loop
- repete enquanto a condição do segundo bloco for verdadeira
- roda o terceiro bloco todo final do loop;
- Equivalente a:
  - int i = 0;
  - while(i <n){
    - ...
    - i++;
  - }