



**DEI**  
DEPARTAMENTO  
DE ENGENHARIA INFORMÁTICA  
TÉCNICO LISBOA

# **Apresentação**

## **Introdução aos Algoritmos e Estruturas de Dados**

IAED, 2021/2022

# Corpo Docente

- **Alameda**
  - Vasco Manquinho (teóricas)
  - Luís Guerra e Silva (5 labs)
  - Daniel Seara (2 labs)
  - Diogo Pacheco (2 labs)
  - Miguel Ramos (2 labs)
  - Leonardo Azevedo (1 lab)
- **Horários de Dúvidas**
  - Afixados na página da cadeira

# Carga Horária

- **Aulas Teóricas:** 5 horas semanais
- **Aulas de Laboratório:** 2 aulas semanais de 1h30
  - Apenas alunos inscritos podem frequentar as aulas de lab.

# Horário (Téóricas & Labs)

	Seg 3/7	Ter 3/8	Qua 3/9	Qui 3/10	Sex 3/11
07:00					
08:00					
09:00	08:30 - 10:00 L LAB 1	08:30 - 10:00 L LAB 3		08:30 - 10:00 L LAB 2	08:30 - 10:00 L LAB 3
10:00		10:00 - 10:00 L LAB 3		10:00 - 10:00 L LAB 4	10:00 - 10:00 L LAB 4
11:00	10:30 - 12:30 T QA02.4	10:30 - 10:00 L LAB 1		10:30 - 10:00 L LAB 4	10:30 - 10:00 L LAB 3
12:00		11:30 - 11:30 L LAB 3		11:30 - 12:30 T QA02.4	11:30 - 11:30 L LAB 4
13:00	12:30 - 13:00 T QA02	13:00 - 14:30 L LAB 5			12:00 - 12:30 T QA02
14:00	14:00 - 15:00 L LAB 2				13:00 - 14:00 T QA02.1
15:00	15:00 - 16:00 L LAB 5				
16:00					
17:00					
18:00					

# Aulas de Laboratório

- O trabalho nas aulas laboratoriais **será individual!**
- Os alunos **têm de comparecer** no turno laboratorial em que se inscreveram
- Inscrições nos laboratórios congeladas no final da 1ª semana
- Aulas divididas em duas partes:
  - Apoio a cada aluno na resolução do guia de laboratório
  - Resolução de uma ficha (em algumas semanas)
- Guia de laboratório disponibilizado antes para **preparação da aula**
- Será possível testar as vossas soluções dos laboratórios no Mooshak
  - Começa no guia do Lab 02
  - Detalhes serão disponibilizados na página

# Projectos

- Os projectos serão realizados **individualmente**
- No final do semestre haverá um **teste prático individual** sobre o projecto. A nota final da componente de projecto depende deste teste prático
- Utilização de sistema automático para entrega de projectos designado de Mooshak (tal como em FP).
  - Mais informação acerca do sistema ao longo do semestre na página da disciplina.
- **Não são aceites entregas por email!**

# Avaliação

- **Fichas de Laboratório:** *média das melhores 4 fichas*
  - 5 fichas **individuais** a realizar nas aulas de laboratório
  - Nota mínima de 8 valores
  - Não há repescagem das fichas
- **Projecto (P):** *2 entregas*; nota mínima de 8 valores; nota de cada aluno ponderada pela nota do teste prático
  - 1ª Entrega: 01 Abril
  - 2ª Entrega: 20 Abril

# Avaliação

## Exame (E)

- Nota mínima de 8 valores

## Nota do Final do Projecto (NP)

- $NP = \min(P, TP) + \sqrt{(|P - TP|/2)}$

## Nota Final ( $NF = 0.3 * F + 0.4 * NP + 0.3 E$ )

- Arredondamentos apenas na nota final

$P = (\text{Projecto1} + \text{Projecto2}) / 2$

TP = Teste Prático



# Avaliação

- Anos Anteriores – Projectos
  - Todos os alunos a frequentar a disciplina para aprovação ou que pretendam fazer melhoria de classificação **serão avaliados em todas as componentes de avaliação**
- Alunos não inscritos
  - Os alunos não inscritos não serão avaliados

# Avaliação

- Não serão tolerados quaisquer tipos de fraude em qualquer componente da avaliação
  - Exemplos:
    - Obtenção do enunciado antes de prova
    - Cópias totais ou parciais de fichas
    - Cópias totais ou parciais de funções/projecto
    - Fornecer código do projecto a outro grupo
    - ...
- Comunicação da tentativa de fraude aos órgãos do IST
- Penalizações serão as maiores possíveis de acordo com o regulamento do IST

# Programa

- Introdução à linguagem de programação C – Parte I
- Introdução ao estudo da eficiência de algoritmos
- Algoritmos de ordenação: inserção directa, selecção directa, bubblesort, quicksort, heapsort e radixsort
- Introdução à linguagem de programação C – Parte II
- Estruturas de dados: pilhas, filas de espera, filas de prioridade, amontoados, árvores e tabelas de dispersão
- Grafos: representação, pesquisa em largura e profundidade, árvores abrangentes de menor custo

# Bibliografia Recomendada



- Programação em C
  - B. Kernighan e D. Ritchie, *The C Programming Language*, 1988, Prentice Hall



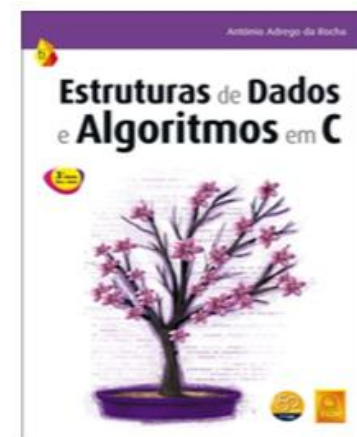
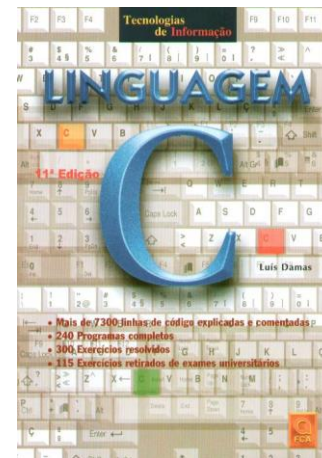
- Estruturas de Dados e Algoritmos de Ordenação
  - R. Sedgwick, *Algorithms in C: Parts 1-4*, 1998, Addison-Wesley Publishing Company



- Grafos
  - T. Cormen, C. Leiserson, R. Rivest, C. Stein, *Introduction to Algorithms*, 2001, McGraw Hill e MIT Press
  - Livro de referência em Análise e Síntese de Algoritmos

# Bibliografia Adicional

- Programação em C
  - Luis Damas, *Linguagem C*, FCA – Editora Informática
- Estruturas de Dados e Algoritmos em C
  - FCA – Editora Informática



# Material de Apoio

- **Teóricas:** slides das aulas teóricas, afixados depois da aula ter decorrido.
- **Laboratórios:** enunciados dos exercícios de laboratório para cada semana (afixados durante a semana anterior, para preparação prévia da aula)
- **Projecto:** enunciados dos projectos, ficheiros de exemplo, etc.
- **Avaliação:** enunciados e resoluções de testes de anos anteriores
- **FAQ**

Ver secção “[Material de Apoio](#)” no site da cadeira

# Dúvidas?



**DEI**

DEPARTAMENTO  
DE ENGENHARIA INFORMÁTICA

TÉCNICO LISBOA

# **Introdução à Linguagem C**

## **K&R: Capítulo 1**

IAED, 2021/2022



# Introdução



- Desenvolvida em 1972 por Dennis Ritchie, nos Bell Labs, para utilização no sistema operativo UNIX
- O standard ANSI C (ISO/IEC 9899:1990) foi adoptado pela ISO em 1990
- Linguagem de alto nível, mas que permite acesso de baixo nível a memória e dispositivos
- A maioria dos sistemas operativos actuais (Linux, Windows, MacOS, etc) continua a ser programado em C
- Influenciou o desenvolvimento de diversas linguagem como: Java, C++, C#, Perl, PHP, JavaScript, etc

# Introdução

- Introdução rápida à linguagem C, utilizando exemplos:
  - Programa que escreve `hello, world`
  - Conversão de temperaturas
  - Cópia de ficheiros
  - Contagem de caracteres
  - Contagem de linhas
  - Contagem de palavras
- Conteúdos
  - Tabelas
  - Funções
  - Passagem por valor
  - Tabelas de caracteres
  - Variáveis externas



# hello, world

```
#include <stdio.h>

main()
{
    printf("hello, world\n");
}
```



# hello, world

```
#include <stdio.h>
```

```
main()  
{  
    printf("hello, world\n");  
}
```

- Biblioteca de funções de entrada/saída

# hello, world

```
#include <stdio.h>
```

```
main()
```

```
{
```

```
    printf("hello, world\n");
```

```
}
```

- Nome de uma função pode ser qualquer
- Todos os programas têm uma função **main**
  - A função `main` é a primeira a ser executada
- Funções podem ser definidas em múltiplos ficheiros  
(a ver mais à frente no semestre)

# hello, world

```
#include <stdio.h>

main()
{
    printf("hello, world\n");
}
```

- Parâmetros formais da função
- Comunicação do exterior com a função
- Neste caso não existem parâmetros



# hello, world

```
#include <stdio.h>

main()
{
    printf("hello, world\n");
}
```

- Instruções associadas a uma função são agrupadas com chavetas

# hello, world

```
#include <stdio.h>

main()
{
    printf("hello, world\n");
}
```

- Nesta função temos apenas uma instrução
  - Chamada à função `printf`
- **Parâmetros** actuais da função `printf` **colocados entre parêntesis**
- **Instruções separadas por ;**



# hello, world

```
#include <stdio.h>

main()
{
    printf("hello, world\n");
}
```

- Cadeia de caracteres entre aspas
- Outros caracteres
  - \n Sequência de caracteres que representa *newline*
  - \t *tab*
  - \b *backspace*
  - \" aspas
  - \\ barra

# hello, world

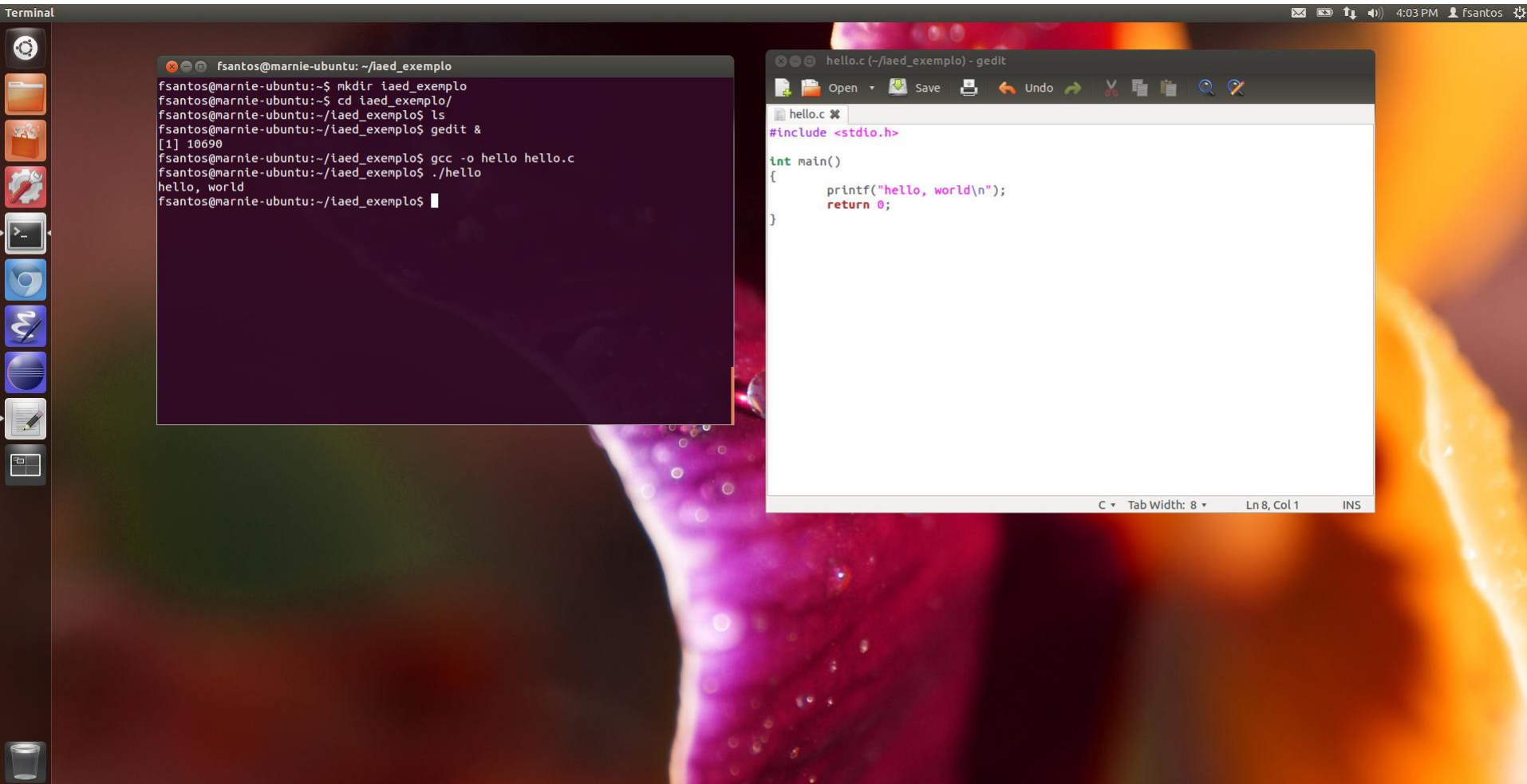
```
#include <stdio.h>

int main()
{
    printf("hello, world\n");
    return 0;
}
```

- Vamos compilar este programa. Como fazer? Onde devo escrever o meu código? Em que sistema operativo?

(já voltaremos aos tipos)

# Ambiente UNIX (e.g. LINUX)



The screenshot displays a Linux desktop environment. On the left, a vertical dock contains icons for the Dash, Home, Applications, and a terminal. The main workspace features two windows. The left window is a terminal titled 'Terminal' with the following commands and output:

```
fsantos@marnie-ubuntu: ~/iaed_exemplo
fsantos@marnie-ubuntu:~$ mkdir iaed_exemplo
fsantos@marnie-ubuntu:~$ cd iaed_exemplo/
fsantos@marnie-ubuntu:~/iaed_exemplo$ ls
[1] 10690
fsantos@marnie-ubuntu:~/iaed_exemplo$ gcc -o hello hello.c
fsantos@marnie-ubuntu:~/iaed_exemplo$ ./hello
hello, world
fsantos@marnie-ubuntu:~/iaed_exemplo$
```

The right window is a code editor titled 'hello.c (~/iaed\_exemplo) - gedit'. It contains the following C code:

```
#include <stdio.h>

int main()
{
    printf("hello, world\n");
    return 0;
}
```

The desktop background is a vibrant, abstract image with purple and orange hues. The system tray at the top right shows the time as 4:03 PM and the user as fsantos.

# Editores de texto

- Editores de texto simples (Unix)
  - Geany (all), gedit (GNOME/Linux), Sublime (all), kate (KDE/Linux), jedit (all), emacs (all), vi (all), textmate (MacOS), Aquamacs (MacOS), Visual Studio Code (all), etc. (pick your own!)
- IDEs (Integrated development environment, Linux, Mac & Windows)
  - Eclipse IDE for C/C++ developers
  - Code::Blocks, etc.

# Linux command line (exemplos)

- \$ **mkdir** iaed\_test (cria uma directoria)
- \$ **cd** iaed\_test (muda de directoria)
- \$ **cd** .. (desce uma directoria)
- \$ **ls** (lista os ficheiros e directorias nessa posição)
- \$ **kate** & ou \$ **gedit** & ou \$ **eclipse** &  
(o & chama os editores e coloca-os a correr em background sem bloquear o terminal)

Ver mais exemplos aqui:  
<http://linuxcommand.org/>

# Linux command line (exemplos)

Compilar um programa com o **gcc**

- \$ **gcc -o hello hello.c** (cria um executável `hello` compilando o código contido no ficheiro `hello.c`)
- \$ **./hello** (correr o executável!)

resultado

```
$ ./hello
```

```
hello, world
```

```
$
```

# Novidades em C

```
def factorial(n):  
    if n == 0:  
        return 1  
    else:  
        return n*factorial(n-1)
```

Python

```
int factorial (int n)  
{  
    if (n == 0) {  
        return 1;  
    }  
    else {  
        return n*factorial(n-1);  
    }  
}
```

C

- **Tipificação estática** (C não permite dinâmica)
  - Tipo de cada variável definido na compilação

# Novidades em C

```
def factorial(n):  
    if n == 0:  
        return 1  
    else:  
        return n*factorial(n-1)
```

Python

```
int factorial (int n)  
{  
    if (n == 0) {  
        return 1;  
    }  
    else {  
        return n*factorial(n-1);  
    }  
}
```

C

- Tipificação estática (C não permite dinâmica)
  - Tipo de cada variável definido na compilação
- Blocos delimitados por { } em vez de indentação
  - Indentação é apenas utilizada para tornar o código mais legível
  - Se o bloco só contém uma instrução os { } são opcionais



# Novidades em C

```
def factorial(n):  
    if n == 0:  
        return 1  
    else:  
        return n*factorial(n-1)
```

Python

Assim também funciona!

```
int factorial (int n)  
{  
    if (n == 0) {  
        return 1;  
    }  
    else {  
        return n*factorial(n-1);  
    }  
}
```

C

- Tipificação estática (C não permite dinâmica)
  - Tipo de cada variável definido na compilação
- Blocos delimitados por { } em vez de indentação
  - Indentação é apenas utilizada para tornar o código mais legível
  - Se o bloco só contém uma instrução as { } são opcionais

# Novidades em C

```
def factorial(n):  
    if n == 0:  
        return 1  
    else:  
        return n*factorial(n-1)
```

Python

```
int factorial (int n)  
{ if (n == 0) {return 1;}  
else {return n*factorial(n-  
1);} }
```

C

Assim também !!!!

- Tipificação estática (C não permite dinâmica)
  - Tipo de cada variável definido na compilação
- Blocos delimitados por { } em vez de indentação
  - Indentação é apenas utilizada para tornar o código mais legível
  - Se o bloco só contém uma instrução as { } são opcionais

# Factorial

```
int factorial (int n)
{
    if (n == 0)
        return 1;
    else
        return n*factorial(n-1);
}
```

- Tipo de dados do valor de retorno da função
  - Precede o nome da função
- Tipos básicos: `char`, `int`, `long int`, `double` e `float`;
- Tipos complexos: estruturas, uniões, tabelas, ponteiros para tipos mais simples e funções

# Factorial

```
int factorial (int n)
{
    if (n == 0)
        return 1;
    else
        return n*factorial(n-1);
}
```

- Tipo de dados do parâmetro  $n$
- Todos os parâmetros são precedidos pelo seu tipo

# Factorial

```
int factorial (int n)
{
    if (n == 0)
        return 1;
    else
        return n*factorial(n-1);
}
```

- **Sintaxe da estrutura de controlo: if-then-else**  
if (<condição>)  
    <instrução1>  
else  
    <instrução2>
- **Se a condição é verdadeira, executa instrução1**
- **Caso contrário, executa instrução2**

# Factorial

```
int factorial (int n)
{
    if (n == 0)
        return 1;
    else
        return n*factorial(n-1);
}
```

- Sintaxe da instrução `return`  
`return <expressão>;`
- Termina a execução da função
- Resultado da `expressão` é o valor de retorno da função

# Trabalho de casa

- Configurar um ambiente de trabalho para a cadeira de IAED nos vossos computadores pessoais. Sugestão:
  - Instalar & configurar uma distribuição Linux (e.g., Ubuntu de modo a ficarem com as mesmas configurações dos laboratórios & compilador *gcc* usado na avaliação dos projectos)
  - Escolher/experimentar editores de texto e/ou IDEs. Comecem por ver os mais simples (geany, gedit, kate, sublime-text, etc).
  - Se por qualquer razão preferir usar outro sistema operativo (e.g., MacOS [OK!], Windows [A evitar!]) procurem encontrar uma forma de usar o *gcc*. No Mac instale o “Xcode” e no Windows usem, por exemplo, o “MinGW/GCC” (aqui a versão do *gcc* será diferente da versão Linux/MacOS).

# Trabalho para horas vagas no IST

antes da vossa 1ª aula prática a sério!

- Experimentar os terminais Linux da RNL
  - Verifique se o seu username & password permite o acesso às áreas Linux. Se não conseguir, faça p.f. uma visita à administração da RNL. Pode significar que alguém se esqueceu de colocar um “tick”.
  - **Experimente** um editor de texto e o terminal.
  - Implemente e corra um **hello.c** à imagem do que fizemos hoje.
  - Tente seguir a folha de exercícios nº1. Na próxima semana começaremos com a folha de exercícios nº2, mas...

P.f. lembre-se que aprender uma nova linguagem exige treino. Infelizmente, as aulas laboratoriais, por si só, não chegam...