

AULA 01

ESTRUTURA DE DADOS

Apresentação da Disciplina

Norton T. Roman & Luciano A. Digiampietri

Disciplina

Objetivo: Familiarizar os alunos com a modelagem e a implementação de diferentes estruturas de dados, bem como os algoritmos para gerenciá-las.

Disciplina

Objetivo: Familiarizar os alunos com a modelagem e a implementação de diferentes estruturas de dados, bem como os algoritmos para gerenciá-las.

Não é um curso de C:

Trata-se de um curso para modelar e gerenciar estruturas de dados em memória principal, usando a linguagem C.

Disciplina

Por que C?

Porque a linguagem deixa a cargo do programador todas as operações necessárias para o gerenciamento das estruturas de dados e permite a manipulação de estruturas e ponteiros de maneira explícita.

A linguagem C

C é uma linguagem de programação compilada, imperativa e procedural, criada em 1972.

É uma das linguagens de programação mais populares e a maioria dos sistemas possui compilador para a linguagem C.

Método de Ensino

Começaremos com noções básicas de C e compararemos com o que já aprendemos em Java.

Em seguida, para cada estrutura de dados a ser vista na disciplina:

- Discutiremos a lógica da estrutura**

- Modelaremos em C**

- Implementaremos as funções para gerenciá-la**

Método de Ensino

Gerenciamento das estruturas:

Inicialização

Inserção de elementos

Exclusão de elementos

Busca por um elemento específico

Contagem do número de elementos

Método de Ensino

Indução ao erro:

Errar é uma das melhores formas de aprender

Implica experimentar e analisar os resultados

Em vários casos apresentaremos programas propositalmente errados, para que o aluno possa ver o comportamento do sistema e identificar a razão do erro

C não é orientada a objetos

É uma linguagem imperativa e estruturada.

Ela permite de maneira mais explícita a alocação e desalocação de memória, gerenciamento de ponteiros e estruturas

Existem outras linguagens inspiradas/baseadas em C que são orientadas a objetos como C++ e C#, mas nesta disciplina trabalharemos apenas com C

Disciplina

A quem se destina o curso?

Ao aluno que já tem noção de programação e quer aperfeiçoar seus conhecimentos sobre modelagem e gerenciamento de estruturas de dados

Quem não sabe programar poderá aprender diversos conteúdos, porém o curso é estruturado pensando naqueles que já têm noção de programação

Material – do que precisaremos?

Um compilador C

O mais conhecido é o GCC para Unix/Linux/Mac (<https://gcc.gnu.org/>) e costuma já estar instalado nas máquinas que utilizam esses sistemas

Para Windows, dois dos mais usados são MinGW (<http://www.mingw.org/>) e LCC

(<https://www.cs.virginia.edu/~lcc-win32/>).

Baixe um para sua plataforma

Material

Usaremos uma IDE?

Integrated Development Environment – ambiente integrado para desenvolvimento de software (editor, compilador, depurador, etc)

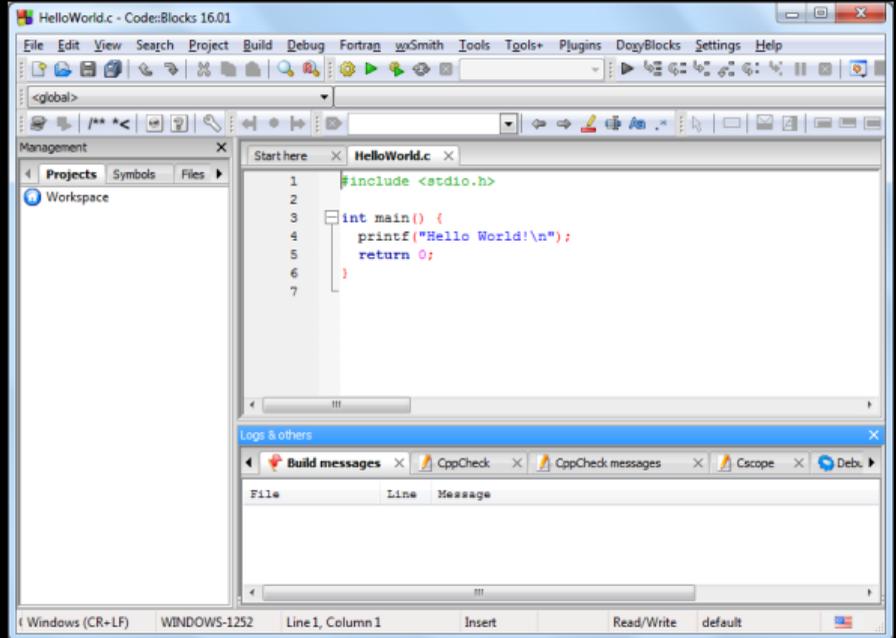
Há várias opções que podem ser usadas...

E qual usar?

Material – IDEs

Code Blocks:

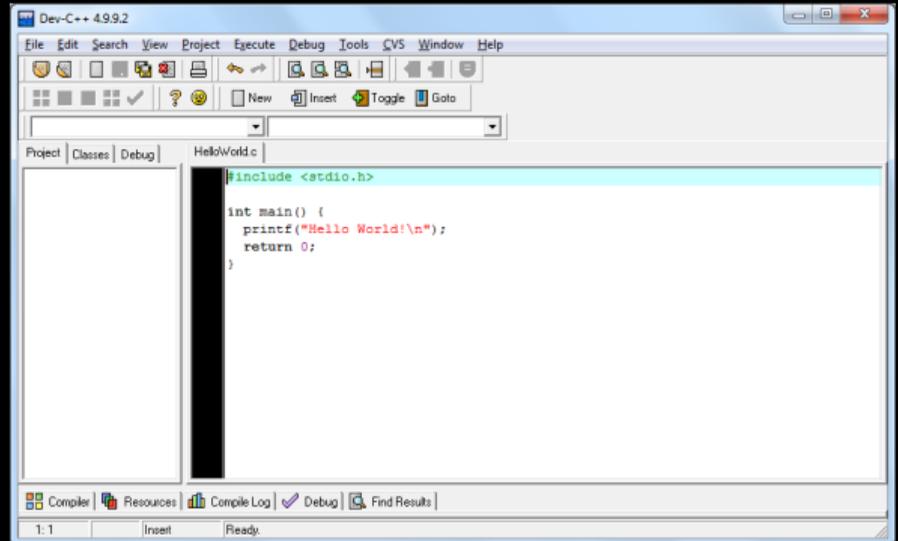
<http://www.codeblocks.org/>



Material – IDEs

Dev-C++:

www.bloodshed.net/devcpp.html

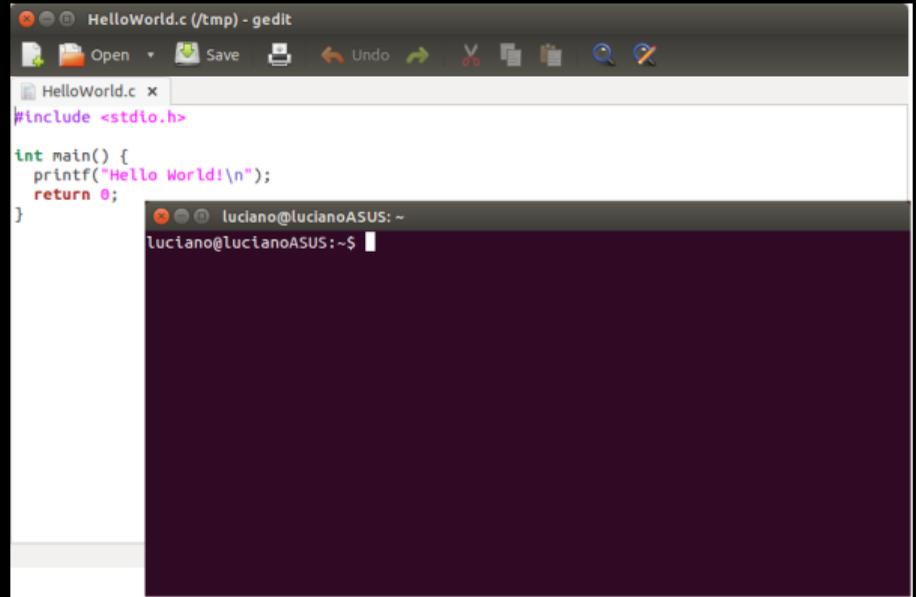


Material – IDEs

E o que
assumiremos que
você tem?

Gedit

Terminal



The screenshot shows a Gedit IDE window titled "HelloWorld.c (/tmp) - gedit". The editor contains the following C code:

```
#include <stdio.h>

int main() {
    printf("Hello World!\n");
    return 0;
}
```

Below the editor, a terminal window is open, showing the prompt "luciano@lucianoASUS: ~" and the command prompt "luciano@lucianoASUS:~\$".

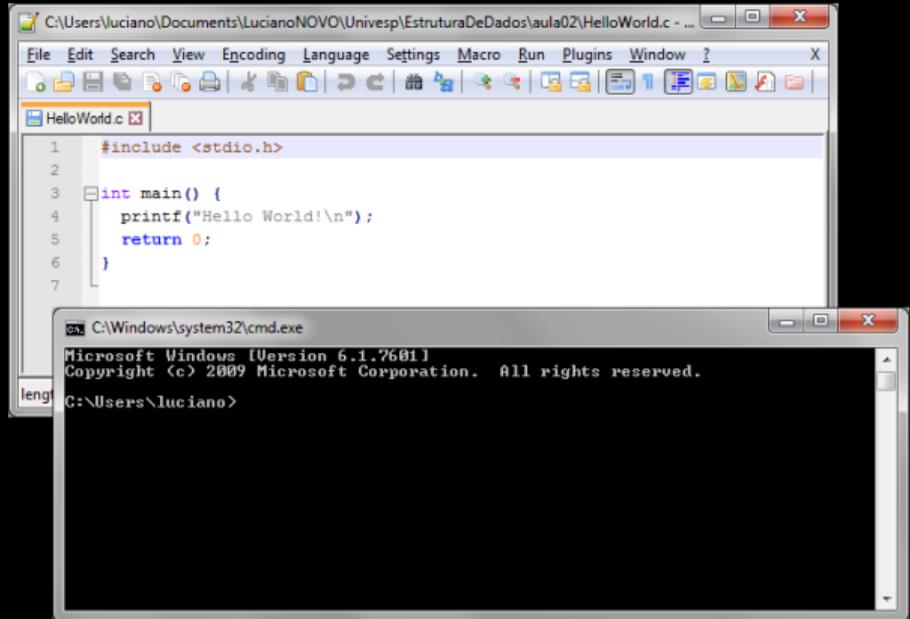
Material – IDEs

E o que
assumiremos que
você tem?

Notepad++

<https://notepad-plus-plus.org/>

cmd



Material – IDEs

Por que não assumiremos uma IDE?

Melhor modo de se entender o que acontece em nosso programa – não há interferência de nada, tentando nos “ajudar”

Não há a carga cognitiva exigida para aprender a usar a IDE

Material de Apoio

Listas de Exercício

Em conjunto com as aulas teremos algumas listas com exercícios

É de extrema importância que sejam feitas, pois acompanham o conteúdo, aumentando o grau de dificuldade a cada exercício

Material de Apoio

Listas de Exercício

Não esqueça que programação não é uma disciplina teórica

Somente a prática faz um bom programador

Um primeiro programa

Um primeiro programa

Em Java:

```
public class HelloWorld {  
    public static void main(String[] args){  
        System.out.println("Hello World!");  
    }  
}
```

Um primeiro programa

Em Java:

```
public class HelloWorld {  
    public static void main(String[] args){  
        System.out.println("Hello World!");  
    }  
}
```

Um primeiro programa

Em Java:

```
public class HelloWorld {  
    public static void main(String[] args){  
        System.out.println("Hello World!");  
    }  
}
```

Um primeiro programa

Em Java:

```
public class HelloWorld {  
    public static void main(String[] args){  
        System.out.println("Hello World!");  
    }  
}
```

Um primeiro programa

Em Java:

```
public class HelloWorld {  
    public static void main(String[] args){  
        System.out.println("Hello World!");  
    }  
}
```

Em C:

Um primeiro programa

Em Java:

```
public class HelloWorld {  
    public static void main(String[] args){  
        System.out.println("Hello World!");  
    }  
}
```

Em C:

```
#include <stdio.h>  
  
int main() {  
    printf("Hello World!\n");  
    return 0;  
}
```

Um primeiro programa

Em Java:

```
public class HelloWorld {  
    public static void main(String[] args){  
        System.out.println("Hello World!");  
    }  
}
```

Em C:

```
#include <stdio.h>  
  
int main() {  
    printf("Hello World!\n");  
    return 0;  
}
```

Um primeiro programa

Em Java:

```
public class HelloWorld {
    public static void main(String[] args){
        System.out.println("Hello World!");
    }
}
```

Em C:

```
#include <stdio.h>

int main() {
    printf("Hello World!\n");
    return 0;
}
```

Um primeiro programa

Em Java:

```
public class HelloWorld {  
    public static void main(String[] args){  
        System.out.println("Hello World!");  
    }  
}
```

Em C:

```
#include <stdio.h>  
  
int main() {  
    printf("Hello World!\n");  
    return 0;  
}
```

Um primeiro programa

Em Java:

```
public class HelloWorld {  
    public static void main(String[] args){  
        System.out.println("Hello World!");  
    }  
}
```

Em C:

```
#include <stdio.h>  
  
int main() {  
    printf("Hello World!\n");  
    return 0;  
}
```

Um primeiro programa

Em Java:

```
public class HelloWorld {  
    public static void main(String[] args){  
        System.out.println("Hello World!");  
    }  
}
```

Como executar?

Em C:

```
#include <stdio.h>  
  
int main() {  
    printf("Hello World!\n");  
    return 0;  
}
```

Um primeiro programa

Em Java:

```
public class HelloWorld {  
    public static void main(String[] args){  
        System.out.println("Hello World!");  
    }  
}
```

Como executar?

```
$ javac HelloWorld.java
```

Em C:

```
#include <stdio.h>  
  
int main() {  
    printf("Hello World!\n");  
    return 0;  
}
```

Um primeiro programa

Em Java:

```
public class HelloWorld {  
    public static void main(String[] args){  
        System.out.println("Hello World!");  
    }  
}
```

Como executar?

```
$ javac HelloWorld.Java  
$ java HelloWorld
```

Em C:

```
#include <stdio.h>  
  
int main() {  
    printf("Hello World!\n");  
    return 0;  
}
```

Um primeiro programa

Em Java:

```
public class HelloWorld {  
    public static void main(String[] args){  
        System.out.println("Hello World!");  
    }  
}
```

Como executar?

```
$ javac HelloWorld.java  
$ java HelloWorld
```

Em C:

```
#include <stdio.h>  
  
int main() {  
    printf("Hello World!\n");  
    return 0;  
}
```

```
$ gcc HelloWorld.c -o HelloWorld
```

Um primeiro programa

Em Java:

```
public class HelloWorld {  
    public static void main(String[] args){  
        System.out.println("Hello World!");  
    }  
}
```

Em C:

```
#include <stdio.h>  
  
int main() {  
    printf("Hello World!\n");  
    return 0;  
}
```

Como executar?

```
$ javac HelloWorld.java  
$ java HelloWorld
```

```
$ gcc HelloWorld.c -o HelloWorld  
$ ./HelloWorld
```

Um primeiro programa

Em Java:

```
public class HelloWorld {  
    public static void main(String[] args){  
        System.out.println("Hello World!");  
    }  
}
```

Em C:

```
#include <stdio.h>  
  
int main() {  
    printf("Hello World!\n");  
    return 0;  
}
```

Como executar?

```
$ javac HelloWorld.java  
$ java HelloWorld
```

```
$ gcc HelloWorld.c -o HelloWorld  
$ ./HelloWorld
```

```
Saída  
$ Hello World!
```

Códigos

Os códigos desta disciplina são baseados na apostila “ACH2023 - Algoritmos e Estruturas de Dados” de Willian Yukio Honda e Ivandré Paraboni

E estão disponíveis em:

<http://www.each.usp.br/digiampietri/ed/>

AULA 01

ESTRUTURA DE DADOS

Apresentação da Disciplina

Norton T. Roman & Luciano A. Digiampietri