Universidade do Minho
Escola de Engenharia

MIETI :: Métodos de Programação II
2015/16

Apresentação

1. Introdução

António Esteves

esteves@di.uminho.pt

Fevereiro 2016

Docente

António Esteves

Departamento de Informática / Gualtar

E-mail: esteves@di.uminho.pt

Tel: 253604481

Web: www.di.uminho.pt/~aje

Horário de atendimento: quinta-feira, 17h-19h, LAP3 do DSI

Objetivos de Ensino

- Em conjunto com a UC de Métodos de Programação I, a presente UC tem por principal objetivo consolidar os conhecimentos sobre **algoritmia** e sua **implementação na linguagem C**.
- São também objetivos da UC:
  - Cobrir **tópicos avançados** da linguagem C
  - Apresentar **estruturas de dados** mais importantes não lecionadas em MP1
  - Descrever e implementar **algoritmos** relevantes para a área das comunicações

Resultados de Aprendizagem

- Ser capaz de **elaborar algoritmos** e de os **implementar** numa linguagem **imperativa** (como o C).
- Conceber e escrever **programas** em C **estruturados** e **modulares**.
- Perceber que a linguagem C é uma **linguagem de nível baixo/intermédio**, e que por isso exige adquirir alguns conhecimentos sobre o *hardware* em que os programas escritos vão ser executados.
- Perceber **estruturas de dados** como listas, pilhas, filas, árvores e grafos, e implementar em C programas que utilizem essas estruturas.

Programa Detalhado

1. Introdução

- Resumo das caraterísticas mais relevantes da linguagem C

2. Tópicos avançados em C

- Utilização de **funções** para dividir um programa em partes menores
- Funções com **múltiplas saídas**/resultados
- Programação **modular**: utilização de ficheiros *header*
- **Alcance** das variáveis
- Declaração de variáveis com *extern*, *static*, *register*
- Utilização de **programas com parâmetros**
- Funções com uma **lista de parâmetros variável**
- Revisão sobre **estruturas**
- Definição de tipos de dados com *typedef*
- Acesso a **ficheiros**
- **Alocação dinâmica** de memória

Programa Detalhado

- Mais sobre apontadores:
  - Apontadores para apontadores
  - *Arrays* de apontadores
  - Apontadores *void*
  - Apontadores para funções.
- Utilização de bibliotecas externas:
  - Símbolos e efetuar a ligação
  - Fazer a ligação com bibliotecas estáticas e dinâmicas
  - Fazer a ligação com bibliotecas externas
  - Questões relacionadas com a “resolução” dos símbolos incluídos no código
  - Criação de bibliotecas.
- Revisitar as bibliotecas normalizadas: stdio.h, ctype.h, stdlib.h, assert.h, stdarg.h, time.h

Programa Detalhado

3. Aspetos de mais baixo nível do C

- Diferença no acesso a ficheiros de texto e binários
- Operações ao nível do bit e conjuntos de bits
- Limites impostos pela representação dos tipos de dados pré-definidos (*int*, *unsigned int*, *float*, *double*, *char*); situações de *overflow*
- Memória física e virtual
- Armazenamento *big endian* vs. little *endian*
- Conversão entre tipos de dados
- Diferença entre pilha e *heap*
- Questões relacionadas com a utilização da pilha em algoritmos recursivos.

Programa Detalhado

4. Definição de estruturas de dados avançadas e respetiva implementação em C

- Listas ligadas
- Listas duplamente ligadas
- Pilhas
- Filas
- Árvores binárias de pesquisa
- Grafos
- Tabelas de *hash*

Programa Detalhado

5. Estruturação e implementação em C de algoritmos importantes na área das comunicações

- Encontrar o caminho mais curto entre dois pontos dum grafo
- Multiplicação de matrizes.

Bibliografia

- António Esteves. *Sebenta de Apoio às Aulas Teóricas de Métodos de Programação II*, 2016.
- Brian Kernighan and Dennis Ritchie. *The C Programming Language*, 2nd Edition, Prentice-Hall, 1988. ISBN: 9780131103627.
- Thomas Cormen, Charles Leiserson, Ronald Rivest, and Clifford Stein. *Introduction to Algorithms*, 2nd Edition, The MIT Press, 2001.
- David Harel. *Algorithmics: The Spirit of Computing*, 3rd Edition, Addison-Wesley, 2004.

Método de Ensino

- Uma sessão teórica de 2H e uma sessão laboratorial de 3H por semana.
- As **sessões teóricas** serão essencialmente para exposição de matéria, demonstração de exemplos no computador e resolução de exercícios em papel.
- Nas **sessões laboratoriais** serão postos em práticas os conhecimentos adquiridos nas sessões teóricas, através da elaboração de algoritmos, escrita de código C e teste do código escrito.
- As sessões laboratoriais servirão ainda para esclarecer dúvidas.
- As **horas de estudo não presencial** deverão ser usadas para preparar as sessões laboratoriais.

Método de Avaliação

- A UC terá dois momentos de avaliação, correspondendo a **dois testes escritos**, um a meio do semestre e outro no final do semestre.
- Cada teste escrito terá um peso de 50% na nota final da UC e será avaliado com uma nota de 0 a 10 valores.
- Não será aprovado na UC, tendo por isso que realizar o exame de recurso, um aluno que reuna alguma das seguintes situações:
  - tenha uma nota inferior a 3.5 (em 10) valores em algum dos testes escritos
  - tenha mais de 1/3 de faltas nas aulas práticas.

Sessões Laboratoriais

Para a realização da componente laboratorial incentiva-se a utilização do seguinte **ambiente de trabalho**:

- O sistema operativo será **Linux**. Sugestão:
  - **Ubuntu 15.10**
- As ferramentas de compilação e depuração serão o **gcc** e **gdb**.
- Poderá usar-se um **IDE** para fazer a interface com o gcc e gdb. Sugestão:
  - **Code::Blocks**
- Recomenda-se levar o portátil pessoal para as aulas laboratoriais