LAB 0: Linguagem C | C language

OBJETIVO: ELABORAR PROGRAMAS EM C E GUARDÁ-LOS NO SOFTWARE DE VERSÕES GITHUB.
Aprender a elaborar programas simples, a criar estruturas de dados simples (pilhas e filas) utilizando vetores e apontadores.

GOAL:TO CREATE C PROGRAMS AND KEEP A VERSION OF THE WORK IN GITHUB OR OTHER VERSION SYSTEM.
To learn how to create simple programs and to create stacks and queues in dynamic and static data strctures.

Desafios para elaboração de programas em C

Pretende-se nos desafios estar alguns conhecimentos em C relacionados com vetores, estruturas de dados, funções, ficheiros, pilhas, filas e apontadores. | The goal of the following challenges are to aquire knowledge in C in the following tasks: vectors, data structures, functions, files, stacks and queues and pointers.

1 Vetor | Vector
- Saber criar um vetor estático de inteiros ou reais e de caracteres. | Create static vectors
- Saber introduzir um elemento no vetor. | To add values in an index of a vector.
- Saber ler o conteúdo de um vetor imprimindo todos os seus elementos no ecrã. | To read the contents of a vector to the screen.
- Saber copiar e concatar vetores de strings (vetor de caracteres). | To know how to cconcatenate and opy vectors of strings.
- Saber criar um vetor dinâmico utilizando apontadores. | Create a dynamic vector using pointers.
- Saber criar espaço na memória para guardar dados num vetor | To create space in memory to a vector.
- Saber imprimir os dados de um vetor dinâmico | To print the data from a dynamic vector.

2 Estruturas | Structures
- Saber criar dados compostos por mais do que um elemento recorrendo a struct. | To know how to create composed data using struct.
- Criar vetor de dados compostos. | Create vectors of composed data.

3 Funções | Functions
- Saber distinguir uma variável passada por valor e passada por referência | To distinguish between by-value and by-reference parameter.
- Criar funções em ficheiros diferentes do ficheiro que contém a função main() e saber compilá-los.

4 Ficheiros | Files
- Abrir e encerrar ficheiros. | Open and close files
- Ler os dados de um ficheiro e imprimi-los no ecrã. | To read data in a file and print them in the screen.

5 Apontadores
.- Saber usar apontadores | To know how to use pointers.

[avançado] 6 Pilha | Stack
- Criar uma pilha recorrendo aos vetores estáticos. | To create a stack using static vector

1.1 Crie um programa que recebe 10 inteiros do teclado utilizando a função gets(), guarda-os num vetor estático de inteiros e imprime os 10 inteiros pela ordem contrária à ordem de entrada.

1.2 Crie um programa que recebe 10 inteiros do teclado utilizando a função gets(), guarda-os num vetor estático de inteiros. Copia esses valores para um outro vetor de float na qual multiplica cada valor por 2.5. Imprime o vetor de floats.

1.3 Crie um programa que recebe uma string, utilizando a função scanf(), utilizando um vetor estático de char. A dimensão da string que recebe tem no máximo 50 caracteres. O programa apresenta o conteúdo da string mas pela ordem inversa.

1.4 Crie dois programas, um que utilize o scanf e outro que utilize o fgets. Em ambos uma cadeia de caracteres é lida através do utilizador. Teste a função ***scanf()*** e a função ***fgets()***.
Tenha em conta que:
a) Qualquer string em C é guardada num vetor e o elemento no final dessa string corresponde a um carácter *null* (o carácter '\0').
b) A função ***scanf*** lê a sequência de caracteres somente até o primeiro carácter espaço. Para ler uma sequência de caracteres com o carácter espaço deve-se usar [%^\n]:

```
scanf("%[^\n]",&s);
```

c) Pode utilizar-se a função ***fgets***.

```
char s[100];
fgets (s, 100, stdin);
printf ("string lida: %s\n", s);
```

1.5 Considere o segunte makefile:

```
#Ficheiro makefile
CC=gcc
CFLAGS=-I.
LIBS = -lm
strings: strings.o strings_main.o
	$(CC) -o strings strings.o strings_main.o $(CFLAGS) $(LIBS)
strings_main.o: strings_main.c
	$(CC) -c -o strings_main.o strings_main.c $(CFLAGS) $(LIBS)
strings.o: strings.c
	$(CC) -c -o strings.o strings.c $(CFLAGS) $(LIBS)
.PHONY: clean
clean:
	rm strings strings_main.o strings.o
```

a) Crie um programa strings.c e strings.h que guardam a função imprime_string(char* s) e que imprime a string s.
b) O programa strings_main.c tem o segunte conteúdo:

```
#include "string.h"
void main(){
  char s[50];
  scanf("%[^\n]",&s);
  imprime_string(s);
  printf("O comprimento da string é: %d\n",strlen(s));
  //The escape character '\0' defines the end of the string. Although invisible
  //it is added by C compiler. In this case it is added to a position of the string.
  s[strlen(s) - 2]='\0';
  imprime_string(s);
  printf("O comprimento da string é: %d\n",strlen(s));
}
```

c) Execute o programa após fazer a compilação e linkagem usando make.
d) Interprete os resultados que obtém:
- Por que razão a segunda vez que imprime a string no ecrã ela é cortada?
- Por que razão o comprimento das strings são diferentes?

1.6 Crie um programa que receba através da função scanf() um conjunto de 4 palavras. Essas palavras devem depois ser concatenadas numa única string, O resultado deve ser apresentado no ecrã.

1.7 Crie um programa que cria um apontador para carácter. Deve criar dinamicamente espaço para uma string de 40 caracteres. O programa recebe a string pelo teclado e imprime essa string no ecrã.

2.1 Uma estrutura em C cria-se utilizando a expressão struct:

```
 struct Alunos {
  int idade;
  char[30] nome;
};
```

Para definir uma variável, há que incluir o nome struct. Por exemplo, no main():
struct Alunos aluno;

Considere o seguinte programa:
a) struct.h

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct alunos{
  int idade;
  char nome[30];
};
```

b) struct.c

```
#include "struct.h"
void main(){
  struct alunos a1;
  strcpy(a1.nome,"João");
  a1.idade = 14;
  printf("O nome é %s e a idade é %d\n",a1.nome,a1.idade);
}
```

Teste o programa e altere-o de forma a incluir a universidade, o nome do curso e o ano que frequenta. Crie um elemento aluno.
Crie um ficheiro makefile para compilar o programa.

2.2 Utilize a estrutura anterior para criar um vetor em que cada elemento é do tipo complexo alunos. Cria um vetor através do comando:
struct alunos vetor_alunos[4];
Crie quatro alunos com nomes e idades diferentes, utilizando o scanf(). Guarde os valores no vetor_alunos. No final imprima no ecrã os valores introduzidos.

3.1 Crie um programa que chama duas funções no main(), uma que tem um parametro inteiro que passa por valor e outra que tem um parametro inteiro que passa por referência. Altere o valor desse parâmetro dentre da função e mostre que só no segundo caso é que a alteração afeta o valor da variável no main().

3.2 Crie um programa que é composto por três ficheiros com código C diferentes:
- O primeiro ficheiro soma.c (e respetivo soma.h) que executa a soma de dois inteiros (somaint) e de dois reais (somafloat).
- O segundo ficheiro multiplicacao.c (e respetivo multiplicacao.h) que executa a multiplicacao de dois inteiros e dois reais (multint e multfloat).
- O terceiro ficheiro que somavet.c (e respetivo somavet.h) uma função somavectint que recebe como parâmetro passado por referência um vetor de inteiros e soma todos os inteiros desse vetor e uma função somavectfloat que faz a mesma coisa agora para um vetor de floats.
- O quarto ficheiro main() que testa os diferentes ficheiros. O nome final do ficheiro designa-se por calculadora e deve realizar a compilação utilizando um ficheiro makefile.

4. Faça um pequeno programa que leia um ficheiro constituído por palavras. Apresente as palavras que foram lidas no ecrã (nota: faça uma pesquisa na Internet para encontrar um programa que faça essa leitura (por exemplo, aqui). Não se esqueça que está em Linux e a localização dos ficheiros obedece à estrutra definida utilizando barras para a direita (“/”).

5.Considere os seguintes apontadores:
int *p;
char * pc;
float* pf;

5.1 Crie um programa que receba do utilizador um inteiro e o guarde numa varável estática *a*. Coloque o apontador p a apontar para a, incremente o conteúdo de p e imprima o valor de *a*. Confirme que o valor de *a* é incrementado por 1.

5.2. Faça o exercício 5.1 para o apontador para float.

5.3 Utilizando o comando malloc() crie espaço para criar um vetor de 10 caracteres apontado por *pc*. Receba do ecrã uma string e guarde essa string no apontador *pc*.

5.4 Utilizando o comando malloc crie dois vetores de inteiros. Peça ao utilizador um número. Teste se esse número é par ou ímpar. Se é par coloque no primeiro vetor, se é ímpar coloque no segundo vetor. Continue a execução até o utilizador introduzir o valor -999 ou quando um dos vetores tiver o valor máximo de elementos.

6 Teste oa programaa em baixo, introduzindo elementos e retirando elementos de uma pilha. Deve acrescentar os comandos no main():

```
#include <stdio.h>

int MAXSIZE = 8;
int stack[MAXSIZE];
int top = -1;

int isempty() {
   if(top == -1)
      return 1;
   else
      return 0;
}
int isfull() {
   if(top == MAXSIZE)
      return 1;
   else
      return 0;
}
int peek() {
   return stack[top];
}
int pop() {
   int data;
   if(!isempty()) {
      data = stack[top];
      top = top - 1;
      return data;
   } else {
      printf("Could not retrieve data, Stack is empty.\n");
   }
}
int push(int data) {
   if(!isfull()) {
      top = top + 1;
      stack[top] = data;
   } else {
      printf("Could not insert data, Stack is full.\n");
   }
}

int main() {
   //test
   return 0;
}
```