DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

Introdução à Programação em C

Funções

IAED

Resumo

- Funções
  - Definição
  - Protótipo e implementação
  - `return` e `void`
  - Passagem por valor
- Exemplos

Funções

- Definição

```
<tipo retorno> <nome> ( <declaracao parametros> )
{
  <declaracoes>
  <instrucoes>
}
```

- Protótipo

```
<tipo retorno> <nome> ( <declaracao parametros> );
```

  - Nos protótipos podemos indicar apenas os tipos de dados nos parâmetros

Funções - instrução `return`

- Permite retornar um valor da função para uma outra função onde foi invocada

```
return <expressao>;
```

- Valor de `<expressao>` convertido para o tipo de retorno da função
- Ao executar a instrução `return`, a função termina de imediato

Exemplo 1: Função Potência

- Objectivo:
  - Escrever uma função que calcule o valor da função potência dados a base e o expoente, números inteiros.

Função Potência

```c
#include <stdio.h>

int potencia(int base, int n)
{
  int i, p = 1;
  for(i = 1; i <= n; i++)
    p = p * base;
  return p;
}

int main()
{
  int i;
  for(i = 0; i < 10; i++)
    printf("%d %d %d\n", i, potencia(2,i), potencia(-3,i));
  return 0;
}
```

Função Potência

protótipo

```c
#include <stdio.h>

int potencia(int base, int n);

int main()
{
  int i;
  for(i = 0; i < 10; i++)
    printf("%d %d %d\n", i, potencia(2,i), potencia(-3,i));
  return 0;
}

int potencia(int base, int n)
{
  int i, p = 1;
  for(i = 1; i <= n; i++)
    p = p * base;
  return p;
}
```

Função Potência

```
#include <stdio.h>

int potencia(int base, int n);

int main()
{
  int i;
  for(i = 0; i < 10; i++)
    printf("%d %d %d\n", i, potencia(2,i), potencia(-3,i));
  return 0;
}

int potencia(int base, int n)
{
  int i, p = 1;
  for(i = 1; i <= n; i++)
    p = p * base;
  return p;
}
```

- Parâmetros formais de função são variáveis locais da função (inacessíveis a partir de outras funções)

Função Potência

```
#include <stdio.h>

int potencia(int base, int n);

int main()
{
  int i;
  for(i = 0; i < 10; i++)
    printf("%d %d %d\n", i, potencia(2,i), potencia(-3,i));
  return 0;
}

int potencia(int base, int n)
{
  int i, p = 1;
  for(i = 1; i <= n; i++)
    p = p * base;
  return p;
}
```

- Instrução `return` especifica valor a retornar
- Função pode não retornar valor (declarada como `void`)

Função Potência

```
#include <stdio.h>

int potencia(int base, int n);

int main()
{
  int i;
  for(i = 0; i < 10; i++)
    printf("%d %d %d\n", i, potencia(2,i), potencia(-3,i));
  return 0;
}

int potencia(int base, int n)
{
  int i, p = 1;
  for(i = 1; i <= n; i++)
    p = p * base;
  return p;
}
```

- Protótipo de uma função: especificação dos tipos dos argumentos e do tipo do valor a retornar

Função Potência

```
#include <stdio.h>

int potencia(int , int );

int main()
{
  int i;
  for(i = 0; i < 10; i++)
    printf("%d %d %d\n", i, potencia(2,i), potencia(-3,i));
  return 0;
}

int potencia(int base, int n)
{
  int i, p = 1;
  for(i = 1; i <= n; i++)
    p = p * base;
  return p;
}
```

- Protótipo de uma função: especificação dos tipos dos argumentos e do tipo do valor a retornar
- Podia ser `int potencia(int, int);`

Função Potência

```
#include <stdio.h>

int potencia(int base, int n);

int main()
{
  int i;
  for(i = 0; i < 10; i++)
    printf("%d %d %d\n", i, potencia(2,i), potencia(-3,i));
  return 0;
}

int potencia(int base, int n)
{
  int i, p = 1;
  for(i = 1; i <= n; i++)
    p = p * base;
  return p;
}
```

- Invocação (chamada) da função `potencia`
- Antes de ser invocada uma função tem que ser conhecida

Função Potência

```
#include <stdio.h>

int potencia(int base, int n);

int main()
{
  int i;
  for(i = 0; i < 10; i++)
    printf("%d %d %d\n", i, potencia(2,i), potencia(-3,i));
  return 0;
}

int potencia(int base, int n)
{
  int i, p = 1;
  for(i = 1; i <= n; i++)
    p = p * base;
  return p;
}
```

- Protótipo da função antes da invocação

Função Potência

```c
#include <stdio.h>

int potencia(int base, int n)
{
  int i, p = 1;
  for(i = 1; i <= n; i++)
    p = p * base;
  return p;
}

int main()
{
  int i;
  for(i = 0; i < 10; i++)
    printf("%d %d %d\n", i, potencia(2,i), potencia(-3,i));
  return 0;
}
```

- Mais uma vez, se a implementação da função for feita antes da invocação, não é necessário o protótipo

Funções:

Passagem *por Valor* & Passagem *por Referência*

- Argumentos são **copiados** para variáveis temporárias quando função é executada
- Função não tem acesso aos argumentos (só às cópias)
- Não os pode alterar
- **Excepção**: se o argumento for uma tabela, não é efectuada a cópia da tabela
  - Se a função alterar o conteúdo da tabela, essa alteração tem efeitos fora do contexto da função

Exemplo: passagem por valor

```
int potencia(int base, int n)
{
  int i, p = 1;
  for(i = 1; i <= n; i++)
    p = p * base;
  return p;
}
```

```
int potencia(int base, int n)
{
  int p;
  for(p = 1; n > 0; n--)
    p = p * base;
  return p;
}
```

- Versão que utiliza menos variáveis
- A modificação do valor da variável `n` não é propagada para fora da função

Exemplo: passagem por valor

```
#include <stdio.h>


void inc(int valor)
{
   valor++;
}

int main()
{
   int i = 0;

   inc(i);

   printf("%d\n", i);
   return 0;
}
```

- Qual o output deste código?

Exemplo: passagem por valor

```
#include <stdio.h>


void inc(int valor)
{
   valor++;
}

int main()
{
   int valor = 0;

   inc(valor);

   printf("%d\n", valor);
   return 0;
}
```

- Qual o output deste código?

Exemplo: passagem por valor

```c
#include <stdio.h>


int inc(int valor)
{
   valor++;
   return valor;
}

int main()
{
   int k = 0;

   k = inc(k);

   printf("%d\n", k);
   return 0;
}
```

- Qual o output deste código?

Exemplo: passagem por referencia

```
#include<stdio.h>

int potencia(int base, int n)
{
  int i, p = 1;
  for(i = 1; i <= n; i++)
    p = p * base;
  return p;
}

int main()
{
  int base, ex;
  scanf("%d%d", &base, &ex);
  printf("%d %d %d\n", base, ex, potencia(base, ex));
  return 0;
}
```

Passagem por referência

- Leitura formatada do input usando `scanf`
- Voltaremos a falar deste assunto mais à frente

Exemplo: passagem por referencia

```
#include<stdio.h>

void inicializar_tabela (int tabela[], int tamanho)
{
   int i;
   for(i = 0; i < tamanho; i++)
      tabela[i]=0;
}

int main()
{
   int vec[100];
   inicializar_tabela(vec, 100);
   return 0;
}
```

Tabelas são sempre passadas por referência, ou seja, podem ser alteradas dentro das funções

- Passagem de tabelas

Exemplo: passagem por referencia

```
#include<stdio.h>

void inicializar_tabela (int tabela[], int tamanho)
{
   int i;
   for(i = 0; i < tamanho; i++)
      tabela[i]=0;
}

int main()
{
   int vec[100];
   inicializar_tabela(vec, 100);
   return 0;
}
```

Desta forma indicamos que a função recebe um vector de inteiros

- Passagem de tabelas

Exemplo: passagem por referencia

```
#include<stdio.h>

void inicializar_tabela (int tabela[], int tamanho)
{
   int i;
   for(i = 0; i < tam      i++)
      tabela[i]=0;
}

int main()
{
   int vec[100];
   inicializar_tabela(vec, 100);
   return 0;
}
```

Quando uma função nada retorna, tal é indicado através de "void"

- Passagem de tabelas

Exemplo 2: Número de Aprovações, Nota Mais Alta e Média de um Turno usando Funções

- Objectivo:
  - Considere-se uma lista de inteiros que denota as notas dos alunos inscritos em **um turno** prático com **25 alunos** (assume-se que o tamanho do turno é fixo e o turno está completo)
  - Ler uma lista de 25 inteiros positivos introduzidos pelo utilizador
  - No fim, mostrar a seguinte informação:
    - Número de aprovações, nota mais alta e média
- Funções:
  - Ler notas do turno para um vector
  - Calcular número de aprovações
  - Calcular nota mais alta
  - Calcular média

Número de Aprovações, Nota Mais Alta e Média de um Turno usando Funções

```c
#include <stdio.h>
#define ALUNOS 25

/* Prototipos das Funcoes */
void lerNotas (int v[], int tamanho);
int calcularAprov (int v[], int tamanho);
int calcularMaisAlta (int v[], int tamanho);
float calcularMedia (int v[], int tamanho);

int main () {
  int notas[ALUNOS];

  lerNotas(notas, ALUNOS);
  printf("Aprovacoes: %d, Mais alta: %d Media: %f\n",
         calcularAprov(notas, ALUNOS),
         calcularMaisAlta(notas, ALUNOS),
         calcularMedia(notas, ALUNOS));
  return 0;
}
/* Implementacao das Funcoes */
```

Número de Aprovações, Nota Mais Alta e Média de um Turno usando Funções

```
void lerNotas (int v[], int tamanho)
{
  int i;
  for (i = 0; i < tamanho; i++)
    scanf("%d", &v[i]);
}

int calcularAprov (int v[], int tamanho)
{
  int i, aprovacoes = 0;
  for (i = 0; i < tamanho; i++)
    if (v[i] >= 10)
      aprovacoes++;
  return aprovacoes;
}
```

- Função `lerNotas` não retorna nada
- Tabela não é copiada; Alteração no conteúdo do vector **tem efeitos fora da função**

Número de Aprovações, Nota Mais Alta e Média de um Turno usando Funções

```
int calcularMaisAlta (int v[], int tamanho)
{
  int i, alta = 0;
  for (i = 0; i < tamanho; i++)
    if (v[i] > alta)
      alta = v[i];
  return alta;
}

float calcularMedia (int v[], int tamanho)
{
  int i, soma= 0;
  for (i = 0; i < tamanho; i++)
    soma += v[i];
  return soma / (float) tamanho;
}
```

- `soma += v[i];` equivale a `soma = soma + v[i];`
- No retorno da função `calcularMedia` há uma conversão explícita de tipos para evitar a divisão inteira

Exemplo 3: Número de Aprovações, Nota Mais Alta e Média num Conjunto de Turnos usando Funções

- Objectivo:
  - Considere-se os **4 turnos práticos** de uma disciplina e 4 listas de inteiros que denotam as notas dos alunos inscritos **em cada um** dos turnos práticos (assume-se que o tamanho do turno é fixo e igual a **25 alunos** e todos os turnos estão completos)
  - **Para cada turno** ler uma lista de 25 inteiros positivos introduzidos pelo utilizador
  - No fim mostrar a seguinte informação **para cada turno**:
    - Número de aprovações, nota mais alta e média
- Consigo usar as funções já criadas no exemplo anterior ??
  - Ler notas do turno para um vector
  - Calcular número de aprovações
  - Calcular nota mais alta
  - Calcular média

Número de Aprovações,Nota Mais Alta e Média num Conjunto de Turnos usando Funções

```
#include <stdio.h>
#define TURNOS 4
#define ALUNOS 25

void lerNotas (int v[], int tamanho);
int calcularAprov (int v[], int tamanho);
int calcularMaisAlta (int v[], int tamanho);
float calcularMedia (int v[], int tamanho);

int main () {
  int i, aprovacoes[TURNOS], alta[TURNOS], notas[TURNOS][ALUNOS];
  float media[TURNOS];

  for (i = 0; i < TURNOS; i++)
    lerNotas(notas[i], ALUNOS);
  for (i = 0; i < TURNOS; i++) {
    aprovacoes[i] = calcularAprov(notas[i], ALUNOS);
    alta[i] = calcularMaisAlta(notas[i], ALUNOS);
    media[i] = calcularMedia(notas[i], ALUNOS);
  }
  for (i = 0; i < TURNOS; i++)
    printf("Turno: %d Aprovacoes: %d, Nota mais alta: %d Media: %f\n",
           i, aprovacoes[i], alta[i], media[i]);
  return 0;
}
```

Vector de vectores...

notas[i] é um vector correspondente ao turno i

Exemplo 4: Ler linhas de texto e mostrar a maior

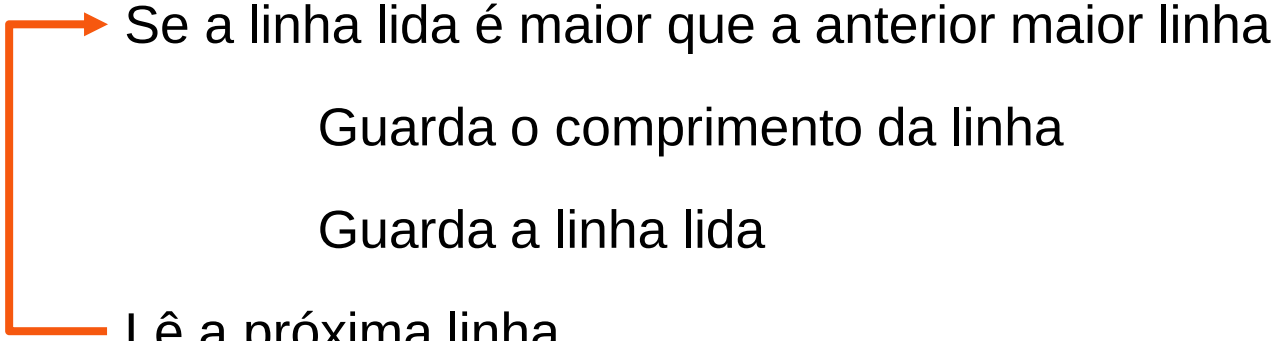
- Objectivo:
  - Ler um conjunto de linhas de texto e escrever a mais comprida (linhas terminam com `'\n'` ou `EOF`)
- Funções:
  - **Ler linha** (que para além de guardar a linha, retorna o seu comprimento)
  - **Copiar linha** : origem ⇨ destino

Ler linhas de texto e mostrar a maior

Algoritmo

Lê linha

Enquanto houver linhas para ler



Se a linha lida é maior que a anterior maior linha

Guarda o comprimento da linha

Guarda a linha lida

Lê a próxima linha

Mostra a maior linha

Ler linhas de texto e mostrar a maior

Algoritmo

```
Lê linha
comprimento = lelinha(linha, MAXLINHA);
Enquanto houver linhas para ler
while (comprimento > 0) {
      Se a linha lida é maior que a anterior maior linha
      if (comprimento > max) {
            Guarda o comprimento da linha
            max = comprimento;
            Guarda a linha lida
            copia(maiscomprida, linha);
      }
      Lê a próxima linha
      comprimento = lelinha(linha, MAXLINHA);
}
Mostra a maior linha
if (max > 0)
      printf("%s", maiscomprida);
```

Ler linhas de texto e mostrar a maior

```
#include <stdio.h>
#define MAX_CHARS 100

int lelinha(char s[], int lim);
void copia(char destino[], char origem[]);

int main() {
  int comprimento, max = 0;
  char linha[MAX_CHARS];
  char maiscomprida[MAX_CHARS];

  comprimento = lelinha(linha, MAX_CHARS);

  while (comprimento > 0) {
    if (comprimento > max) {
      max = comprimento;
      copia(maiscomprida, linha);
    }
    comprimento = lelinha(linha, MAX_CHARS);
  }
  if (max > 0)
    printf("%s\n", maiscomprida);
  return 0;
}
```

Vou lendo para esta string (o meu buffer)

Aqui vou guardando a maior das linhas lidas até ao momento

Ler linhas de texto e mostrar a maior

```
int lelinha(char s[], int lim)
{
  int c, i;

  for (i = 0; i < lim-1 && (c=getchar())!= EOF && c != '\n'; i++)
    s[i] = c;

  s[i] = '\0';
  return i;
}

void copia(char destino[], char origem[])
{
  int i;
  for(i = 0; origem[i] != '\0'; i++)
    destino[i] = origem[i];
  destino[i] = '\0';
}
```

- Para se copiar uma tabela **não basta igualar as variáveis**
- É necessário copiar cada elemento da tabela um a um

Exercício (1)

- Defina uma função que recebe um vector de inteiros e devolve o maior valor no vector.

```
int maior(int vec[], int vec_size)
{









}
```

Exercício (2)

- Defina uma função que recebe uma string e substitui as letras minúsculas por letras maiúsculas.

```
void substituiMinusculas(char str[])
{




}
```

Tabela ASCII

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Imagem retirada de:

http://www.cdrummond.qc.ca/cegep/informat/Professeurs/Alain/files/ascii.htm

Algarismos

letras

LETRAS