

Objetivo: este laboratório tem como objetivo familiarizar-se com a ferramenta Dev-C++ e a forma de escrever, compilar e executar um programa. Irá aprender igualmente aspectos básicos do C. Finalmente, realizará um conjunto de exercícios de código C, explorando tipos de variáveis e usando as funções de leitura e escrita. OS exercícios em C deverão ser submetidos no Classroom, dentro de um ZIP com o nome *PI_Lab3_NomeApelido.zip*


Introdução ao C e ao Dev-C++



Instale a aplicação Dev C++ (já vem com um compilador C, não precisando instalá-lo à parte). Implemente agora o pequeno e famoso código do “Hello world”:

- Crie uma pasta com o nome **Lab3**.
- Abra a aplicação Dev C++.
- Crie um ficheiro novo intitulado **Lab3_ex1.c**, com extensão **.c** (e não **.cpp!**), e grave na pasta Lab2.
- Redija o seguinte código fonte C:

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

Sugestões:

- Na escrita de código, respeite sempre a indentação (colocação de novas linhas e texto “avançado”).
- Avance o texto com a tecla Tab , nunca com espaços!
- Não utilize caracteres com acentos ou “ç”, pois não são reconhecidos pelo compilador.
- Atenção a copiar código escrito em processadores de texto. Muitas vezes código editado em processadores de texto ficam com as aspas desformatadas, em vez de " " fica " ". Exemplo: `printf("Hello World!\n");` é diferente de `printf(“Hello World!\n”);`

- e) **Compile** o programa, premindo na tecla F9 ou no botão .
- O compilador converterá o ficheiro fonte com instruções na linguagem C (ficheiro Lab2_ex1.c) num ficheiro com instruções em código máquina (ficheiro **Lab2_ex1.exe**), de extensão exe, um executável que o seu computador sabe executar.
 - Abra a pasta Lab2, onde encontrará agora um novo ficheiro Lab2_ex1.exe, o programa executável que foi criado.
- f) **Execute** o seu programa Lab2_ex1.exe, premindo na tecla F10 ou no botão .
- Aparecerá uma janela DOS onde aparecerá a frase "Hello World!" que mandou imprimir.
 - Feche sempre a janela de DOS. Se não fechar, o Dev C++ não conseguirá correr novamente.
 - Abra a pasta Lab2, onde encontrará o ficheiro Lab2_ex1.exe. Clique duas vezes no ficheiro, para o executar novamente. Este aparecerá e desaparecerá rapidamente, pois imprime a frase e termina logo a seguir, desaparecendo. Só executado a partir do Dev C++ é que fica à espera que o termine.

Comentários

Os comentários no código servem principalmente para documentação do programa. São ignorados pelo compilador, portanto não irão afectar o programa executável gerado. Existem duas formas de escrever comentários:

- Os comentários de linha iniciam-se com o símbolo `//` e terminam no final da mesma linha.
- Os blocos de comentários iniciam-se com o símbolo `/*` e estendem-se até aparecer o símbolo `*/`. Um bloco de comentário pode aparecer em qualquer lugar no programa onde possa aparecer um espaço em branco e pode estender-se por mais de uma linha.

```
#include <stdio.h>

/* Um comentário de bloco permite
   Os comentários se estendam por
   várias linhas de comentários */

int main (void)
{
    printf("Hello World!\n"); // comentário de linha
    return 0;
}
```

Tipos de Dados

Existem vários tipos de dados, tal como identificado na tabela em baixo e descritos a seguir.

Tipo de dados	Significado	Tamanho (em bits)	Tamanho (em bytes)	Intervalo de valores aceites
char	Caractere	8	1	-128 a 127
short	Inteiro curto	16	2	-32768 a 32767
int	Inteiro	16 ou 32	2 ou 4	-32768 a 32767 ou -2147483648 a 2147483647
long	Inteiro longo	32	4	-2147483648 a 2147483647
float	Real	32	4	3.4E-38 a 3.4E+38
double	Real duplo	64	8	1.7E-308 a 1.7E+308

int: O tipo de dados int (inteiro) serve para armazenar valores numéricos inteiros.

Existem vários tipos de inteiros, cada um de um tamanho diferente (dependendo do sistema operativo e/ou arquitetura do processador):

- **int** - pode possuir 16 bits, 32 bits ou 64 bits
- **short int** - deve possuir tamanho de no mínimo 16 bits e não pode ser maior que int
- **long int** - deve possuir tamanho mínimo de 32 bits

Todos estes tipos de inteiros podem ainda ser declarados precedidos da cláusula **unsigned**, o que faz com que só suporte números positivos. Isto faz com que, com o mesmo tamanho, uma variável suporte mais números positivos do que um signed (todos os inteiros são signed por omissão).

char: O tipo char ocupa 1 byte, e serve para armazenar caracteres ou inteiros (são equivalentes, pela tabela de ASCII, que se mostra de seguida). Isso significa que o programa reserva um espaço de 8 bits para armazenar um valor.

Tabela ASCII

ASCII control characters			ASCII printable characters			Extended ASCII characters										
00	NULL	(Null character)	32	space	64	@	96	`	128	Ç	160	á	192	Ł	224	Ó
01	SOH	(Start of Header)	33	!	65	A	97	a	129	ú	161	í	193	ł	225	ô
02	STX	(Start of Text)	34	"	66	B	98	b	130	ê	162	ó	194	ŀ	226	õ
03	ETX	(End of Text)	35	#	67	C	99	c	131	ä	163	ü	195	ŧ	227	ö
04	EOT	(End of Trans.)	36	\$	68	D	100	d	132	å	164	ñ	196	ı	228	ø
05	ENQ	(Enquiry)	37	%	69	E	101	e	133	à	165	ñ	197	ı	229	õ
06	ACK	(Acknowledgement)	38	&	70	F	102	f	134	â	166	°	198	Ā	230	μ
07	BEL	(Bell)	39	'	71	G	103	g	135	ç	167	°	199	Ă	231	þ
08	BS	(Backspace)	40	(72	H	104	h	136	ê	168	¿	200	Ľ	232	Ɔ
09	HT	(Horizontal Tab)	41)	73	I	105	i	137	ë	169	©	201	Ť	233	ú
10	LF	(Line feed)	42	*	74	J	106	j	138	è	170	¬	202	Ŧ	234	û
11	VT	(Vertical Tab)	43	+	75	K	107	k	139	í	171	½	203	Ţ	235	ü
12	FF	(Form feed)	44	,	76	L	108	l	140	î	172	¾	204	Ŧ	236	ý
13	CR	(Carriage return)	45	-	77	M	109	m	141	ï	173	ı	205	Ŧ	237	ÿ
14	SO	(Shift Out)	46	.	78	N	110	n	142	À	174	«	206	Ŧ	238	·
15	SI	(Shift In)	47	/	79	O	111	o	143	Á	175	»	207	Ŧ	239	ˆ
16	DLE	(Data link escape)	48	0	80	P	112	p	144	Â	176	ˆ	208	Ŧ	240	≡
17	DC1	(Device control 1)	49	1	81	Q	113	q	145	æ	177	ˆ	209	Ŧ	241	±
18	DC2	(Device control 2)	50	2	82	R	114	r	146	Æ	178	ˆ	210	Ŧ	242	ˆ
19	DC3	(Device control 3)	51	3	83	S	115	s	147	ø	179	ˆ	211	Ŧ	243	¼
20	DC4	(Device control 4)	52	4	84	T	116	t	148	ö	180	ˆ	212	Ŧ	244	¶
21	NAK	(Negative acknowl.)	53	5	85	U	117	u	149	õ	181	ˆ	213	Ŧ	245	§
22	SYN	(Synchronous idle)	54	6	86	V	118	v	150	ù	182	ˆ	214	Ŧ	246	÷
23	ETB	(End of trans. block)	55	7	87	W	119	w	151	û	183	ˆ	215	Ŧ	247	ˆ
24	CAN	(Cancel)	56	8	88	X	120	x	152	ÿ	184	©	216	Ŧ	248	ˆ
25	EM	(End of medium)	57	9	89	Y	121	y	153	Ö	185	ˆ	217	Ŧ	249	ˆ
26	SUB	(Substitute)	58	:	90	Z	122	z	154	Ü	186	ˆ	218	Ŧ	250	ˆ
27	ESC	(Escape)	59	;	91	[123	{	155	ø	187	ˆ	219	Ŧ	251	ˆ
28	FS	(File separator)	60	<	92	\	124		156	£	188	ˆ	220	Ŧ	252	ˆ
29	GS	(Group separator)	61	=	93]	125	}	157	Ø	189	ˆ	221	Ŧ	253	ˆ
30	RS	(Record separator)	62	>	94	^	126	~	158	×	190	ˆ	222	Ŧ	254	ˆ
31	US	(Unit separator)	63	?	95	_			159	f	191	ˆ	223	Ŧ	255	nbsp
127	DEL	(Delete)														

Por exemplo, o carater 65 é a letra 'A'. Assim, tendo a variável

```
char carater;
```

As duas instruções seguintes são equivalentes:

```
carater = 65;
```

```
carater = 'A';
```

De mesmo modo é possível fazer:

```
carater = 'A' + 2;
```

cujo valor resultante será 'C'.

float: O tipo de dados float serve para armazenar números de “ponto flutuante” (números reais), ou seja, com casas decimais.

double: O tipo de dados double serve para armazenar números de “ponto flutuante” (números reais) de dupla precisão, normalmente tem o dobro do tamanho do float e portanto o dobro da capacidade.

Diretiva #include

A diretiva #include inclui o conteúdo de um outro arquivo dentro do programa atual, ou seja, a linha que contém a diretiva é substituída pelo conteúdo do arquivo especificado.

O ficheiro stdio.h contém as funções de entrada e saída standard. Para que o compilador “conheça” as funções *printf* e *scanf* é necessário que o programa tenha a seguinte diretiva:

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

Sequência de escape '\'

Uma sequência de escape é uma combinação de caracteres representada por um caracter “backslash” (\) seguido por uma letra ou por combinações de dígitos. É usada na representação de caracteres de controle como CR, apóstrofe, etc.. As sequências de escape mais utilizadas em programação são:

- \n - LF NewLine
- \t - HT Horizontal Tabulation
- \' - Apóstrofo (Single quotation mark)
- \" - Aspas (Double quotation mark)
- \\ - backslash

A função printf()

A função printf serve para enviar dados para a saída standard que, por defeito, está direcionada para o monitor. Como um exemplo simples de utilização do printf temos o seguinte programa:

```
#include <stdio.h>
int main() {
    printf("Isto vai aparecer no monitor\n");
    return 0;
}
```

No exemplo apresentado, a função printf recebe apenas uma string de formato como argumento ("Isto vai aparecer no monitor\n"). O \n é uma sequência de escape que indica que é para mudar de linha. O printf poderá receber mais argumentos os quais serão inseridos na string de formato nas localizações assinaladas com o carácter %.

```
#include <stdio.h>
int main() {
    int custo=10;
    printf("Valor a pagar = %d Euro\n", custo);
    return 0;
}
```

Se executarmos o programa, aparecerá no monitor: **Valor a pagar = 10 Euro**. O que acontece é que o %d é utilizado como um identificador de formato para o próximo argumento (custo). Neste caso é esperado um inteiro. A tabela seguinte apresenta alguns dos especificadores de formato que podemos utilizar na função printf.

Especificador de formato	Tipo
%c	caractere
%d	decimal, inteiro
%e	notação científica com "e" minúsculo
%E	notação científica com "e" maiúsculo
%f	real
%lf	real duplo
%s	string
%%	caracter '%'

Largura do campo e Precisão

Como o próprio nome indica, especifica qual a largura mínima do campo. Se pretendemos especificar o número mínimo de caracteres para apresentar um valor, deveremos indicar um número para a largura do campo.

O campo é impresso de acordo com as seguintes regras:

- Se o valor for mais largo que o campo, este será expandido para poder conter o valor. O valor nunca será cortado.
- Se o valor for menor que o campo, a largura do campo será preenchida com espaços ou zeros. Os zeros são especificados pela opção 0, que precede a largura
- O alinhamento padrão é à direita. Para se alinhar um número à esquerda usa-se a opção "-" (hífen ou sinal de menos) antes da largura do campo.

```
printf ("%5d", 15);    // mostra "   15"
printf ("%05d", 15);   // mostra "00015"
printf ("% -5d", 15);  // mostra "15  "
```

A precisão pode ter quatro significados diferentes:

- Se a conversão solicitada for **inteira**: número mínimo de dígitos a exibir (será preenchido com zeros se necessário).
- Se a conversão for **real**: o número de casas decimais a exibir. O valor será arredondado se a precisão especificada no formato for menor que a do argumento.
- Se a conversão for em **notação científica**: o número de algarismos significativos. O valor será arredondado se o número de algarismos significativos pedido for maior que o do argumento.
- Se a conversão for de uma **sequência de caracteres** (s): o número máximo de caracteres a exibir.

Assim como a largura do campo, a precisão pode ser especificada diretamente por um número ou com um asterisco, mas deve ser precedida por um ponto. É possível combinar a largura com a precisão. Por exemplo, %10.4f indica um campo de número real de comprimento total dez e com 4 casas decimais. Note que, na largura do campo está incluído o ponto decimal.

A função scanf()

A função scanf() é utilizada para a entrada de dados. Para ler um item deve-se passar no primeiro argumento uma string que representa o formato da entrada e nos restantes argumentos os endereços das variáveis de destino. Na string que representa o formato da entrada ("**texto entre aspas**") podemos utilizar os especificadores de formato apresentados anteriormente (%d, %f, %c, %s). Para representarmos o endereço de uma variável colocamos o carácter & antes do identificador da variável.

Código	Descrição	Exemplo de Entrada
<code>scanf("%d",&num);</code>	Lê um valor inteiro que guarda na variável num.	➤ 9
<code>scanf("%d %f", &x1 ,&x2);</code>	Lê um valor inteiro que guarda na variável x1 e em seguida um valor real que guarda na variável x2 do tipo float	➤ 7 9.1234
<code>scanf("%c",&letra);</code>	Lê um carácter que guarda na variável letra.	➤ L
<code>scanf("%s",nome);</code>	Lê um conjunto de caracteres que guarda na variável nome	➤ Luís

Avaliação de expressões

Os operadores binários atuam sobre operandos do mesmo tipo e o tipo do resultado é determinado pelo tipo dos operandos. Se tivermos uma adição de dois inteiros do tipo int o resultado é um int (2+3=5).

Para avaliar expressões poderá ser necessário realizar conversões automáticas de tipos. Se tivermos uma adição de um double com um int, a operação é realizada convertendo o int para double (2.0+3=5.0).

short -> int -> float -> double

A prioridade dos operadores determina a ordem de avaliação das expressões e o que estiver entre parênteses é avaliado em primeiro lugar.

<i>Grau de Precedência</i>	<i>Operação</i>
Alto	()
Médio	-, + (unários)
	*, /, %
Baixo	+, -

Operações sobre inteiros

<i>Operação</i>	<i>Símbolo</i>	<i>Exemplo</i>	<i>Resultado</i>
Adição	+	26 + 10	36
Subtração	-	26 - 1	25
Multiplicação	*	26 * 10	260
Divisão	/	26 / 10	2
Resto da divisão	%	26 % 10	6

Operações sobre reais

<i>Operação</i>	<i>Símbolo</i>	<i>Exemplo</i>	<i>Resultado</i>
Adição	+	5.4 + 2.0	7.4
Subtração	-	5.4 - 2.0	3.4
Multiplicação	*	5.4 * 2.0	10.8
Divisão	/	5.4 / 2.0	2.7

Verdadeiro ou Falso

Como a Linguagem C não possui um tipo lógico, os tipos inteiros são utilizados para representar valores lógicos.

O falso é representado pelo valor zero. Verdadeiro é representado por um inteiro diferente de zero.

Exercícios

Guarde cada exercício no formato IAED_NomeApelido_Lab3_ex1.c. Havendo alterações pedidas ao programa nalguma alínea, grave também usando incluindo no final do nome a alínea (IAED_NomeApelido_Lab3_ex1b.c). Escreva dentro do ficheiro no final, em modo de comentário, as explicações que são pedidas.

Exercício 1

O programa seguinte usa o tipo de dado elementar short:

```
#include <stdio.h>
int main()
{
    short valor= 32;
    int tamanho;

    tamanho = sizeof(valor);

    printf("Um short int : %d\n",valor);
    printf("espaco de memoria ocupado pela variavel: %d bytes", tamanho);
    return 0;
}
```

- Compile e execute o programa. O que aparece escrito no monitor?
- Edite o programa e altere o valor 32 para um outro valor inteiro relativamente baixo, digamos 100. Compile e corra o programa.
- Altere o valor para 90000 e tente compilar o programa. O que é que acontece? Porquê?
- Edite o programa e altere a palavra "short" para "int". Compile e corra o programa. Qual a diferença entre esta e a alínea anterior?

Exercício 2

O programa seguinte usa o tipo de dado elementar float:

```
#include <stdio.h>
int main()
{
    float valor= 900.25;
    int tamanho;

    tamanho = sizeof(valor);
    printf("Um float : %f",valor);
    printf("espaco de memoria ocupado: %d bytes", tamanho);
    return 0;
}
```

- a) Compile e execute o programa. O que aparece escrito no monitor?
- b) O que irá aparecer no monitor se alterar o especificador de formato de %f para %E.
- c) E se alterar para %e?

Exercício 3

O programa seguinte usa o tipo de dados elementar char:

```
#include <stdio.h>
int main()
{
    char ch = 'A';
    printf("Um char : %c\n", ch);
    printf("Outro char : %c\n", ch + 32);
    printf("O carater %c tem o codigo ASCII %d\n", 100, 100);
    return 0;
}
```

- a) Compile e execute o programa. O que aparece escrito no monitor?
- b) Troque o 'A' por 'Z' compile e corra o programa.
- c) O que irá aparecer no monitor se alterar o especificador de formato de %c para %d. (Nota: o código ASCII do carácter 'Z' é 90)

Exercício 4

Para o programa seguinte:

```
#include <stdio.h>
int main()
{
    Char c = 'r'
    short j = 127;
    int k 32767;

    printf("c= %c\n", c);
    c=C+1;
    Printf("c= %c\n, c);
    c=c+1;
    printf("c= %c\n", c);
    printf("j= %d\n", j);
    j=j-1;
    printf("j= %d\n", j);
    j++;
    printf("j= %d\n", j);
    printf("k= %d\n", k);
    k -=4;
}
```

```
printf("k= %d\n", k);  
k = k+5;  
printf("k= %d\n", k);  
printf("Valores finais:\n\tc = %c\n\tj = %d",c,j);  
printf("\n\tk = %d\n\n", k);  
printf("y= %d\n", y);  
return 0;  
}
```

Corrija os erros sintáticos do programa. Lembre-se que variáveis são sensíveis às maiúsculas, cada instrução deve terminar com “;”, e dentro do printf a expressão a imprimir deve estar entre aspas.

Exercício 5

Examine o programa seguinte:

```
#include <stdio.h>  
int main()  
{  
    int notaTeste1 = 15, notaTeste2 = 15, notaTeste3 = 17;  
    double media;  
  
    media = (notaTeste1 + notaTeste2 + notaTeste3)/3;  
    printf("Nota final : %f valores\n", media);  
    return 0;  
}
```

- Compile, corra o programa e examine a sua saída. Concorde com o resultado? O que terá acontecido?
- Substitua a instrução do cálculo da média pela instrução seguinte e observe o que é escrito pelo programa.

```
media = (notaTeste1 + notaTeste2 + notaTeste3)/3.0;
```
- Na instrução printf substitua %f por %3.0f e observe o resultado.
- Na instrução printf substitua %f por %3.2f e observe o resultado
- Altere o programa de modo a que as notas dos testes sejam inseridas pelo utilizador.