

Algoritmos e Estruturas de Dados I

Breve revisão da linguagem C

Profa. Teoria: Mirtha Lina Fernández Venero, Sala 529-2,

mirtha.lina@ufabc.edu.br

<http://professor.ufabc.edu.br/~mirtha.lina/aedi.html>

Prof. Prática: Paulo Henrique Pisani, Sala 507-2

paulo.pisani@ufabc.edu.br

<http://professor.ufabc.edu.br/~paulo.pisani/2019Q1/AEDI/index.html>

11 de fevereiro de 2019

Atenção: Para assinar a presença de hoje, preencha o formulário disponível em

<https://goo.gl/forms/khucA4AAdeWfr1OU2>

Agenda

Introdução

Elementos básicos da linguagem C

Ambiente de trabalho

Vetores e Strings

Funções

Estudo independente

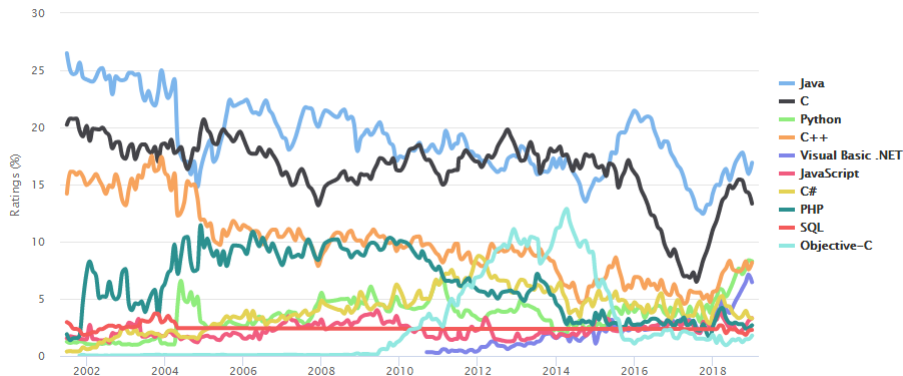
Exercícios para casa

Bibliografia

Linguagem C desenvolvida por Dennis Ritchie, 1969-73

TIOBE Programming Community Index

Source: www.tiobe.com



“C is quirky, flawed, and an enormous success. C++ and Java, say, are presumably growing faster than plain C, but I bet C will still be around.”

Dennis Ritchie's quotes



Algumas características da linguagem C

- ▶ linguagem estruturada, flexível, modular, de propósito geral, combina características de alto e baixo nível, referência para outras linguagens e.g. C++, Java, C#, Go, Arduino, Verilog
- ▶ variedade de tipos de dados, operadores e funções predefinidas
- ▶ *lingua franca*: livros didáticos, ..., desenvolvedores
- ▶ linguagem portátil, com compiladores para todos os sistemas operacionais e plataformas de hardware
- ▶ compilador que gera programas **muito** eficientes

	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34



Algumas características da linguagem C

- ▶ linguagem estruturada, flexível, modular, de propósito geral, combina características de alto e baixo nível, referência para outras linguagens e.g. C++, Java, C#, Go, Arduino, Verilog
 - ▶ variedade de tipos de dados, operadores e funções predefinidas
 - ▶ *lingua franca*: livros didáticos, ..., desenvolvedores
 - ▶ linguagem portátil, com compiladores para todos os sistemas operacionais e plataformas de hardware
 - ▶ compilador que gera programas **muito** eficientes
- +/- ponteiros e funções para o gerenciamento de memória
- não é orientada a objetos (classes, encapsulation, data hiding, inheritance, polymorphism, constructors/destructors, genericity, runtime type checking, etc)

“Anybody who comes to you and says he has a perfect language is either naive or a salesman.” [Bjarne Stroustrup's quote](#)



Agenda

Introdução

Elementos básicos da linguagem C

Ambiente de trabalho

Vetores e Strings

Funções

Estudo independente

Exercícios para casa

Bibliografia



Tipos de Dados Básicos: Inteiros e Reais

Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295
float	4 byte	1.2E-38 to 3.4E+38
double	8 byte	2.3E-308 to 1.7E+308
long double	10 byte	3.4E-4932 to 1.1E+4932

Tipos de dados simples, constantes e variáveis

- ▶ Além dos inteiros, incluem o tipo `char` e `void` que indica nenhum valor disponível;
- ▶ não existe tipo `bool`: qualquer valor diferente de zero é **true** enquanto zero representa **false**;

Declaração de variáveis: 1) tipo 2) identificadores 3) inicialização

```
#define MPI 3.14 // Constant declaration

int main(void) {

    const int LENGTH = 10; // Constant declaration
    int i, j = 2, k; // Variable declaration
    float minSalary = 954.0;
    char carr = 'H';

}
```


Identificadores e palavras reservadas

As palavras reservadas de C não podem ser usadas como identificadores; as funções predefinidas também não.

auto	else	long	switch
break	enum	register	typedef
case	extern	return	union
char	float	short	unsigned
const	for	signed	void
continue	goto	sizeof	volatile
default	if	static	while
do	int	struct	_Packed
double			

Use identificadores descritivos nem muito curtos nem muito longos, começando por minúscula. Sinta-se a vontade de usar um estilo próprio porém consistente que ajude na legibilidade do seu código!

Saída e Entrada de dados, stdio.h

en.cppreference.com/w/c/io/fprintf

Defined in header <stdio.h>

```
int printf( const char *restrict format, ... );
int fprintf( FILE *restrict stream, const char *restrict format, ... );
int sprintf( char *restrict buffer, const char *restrict format, ... );
```

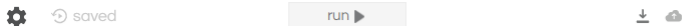
en.cppreference.com/w/c/io/fscanf

Defined in header <stdio.h>

```
int scanf( const char *restrict format, ... );
int fscanf( FILE *restrict stream, const char *restrict format, ... );
int sscanf( const char *restrict buffer, const char *restrict format, ... );
```

<i>specifier</i>	Output
<i>d or i</i>	Signed decimal integer
<i>f</i>	Decimal floating point, lowercase
<i>c</i>	Character
<i>s</i>	String of characters
<i>p</i>	Pointer address
<i>%</i>	A % followed by another % character will write a single % to the stream

Exemplo de Entrada e Saída de dados



```

main.c
1 #include <stdio.h>
2
3 int main(void) {
4     int i, j; float x, y; char str1[10], str2[4]; char carr[2];
5
6     char input[] = "25 54.32E-1 Thompson 56789 0123 56ab";
7
8     int ret = sscanf(input, "%d%f %9s %2d %f %*d %3[0-9] %2c",
9                    &i,&x, str1, &j, &y, str2, carr);
10    /* parse as follows:
11       |   %d: an integer           => i
12       |   %f: a floating-point value => x
13       |   %9s: a string of at most 9 non-whitespace characters => str1
14       |   %2d: two-digit integer (digits 5 and 6) => j
15       |   %f: a floating-point value (digits 7, 8, 9) => y
16       |   %*d: an integer which isn't stored anywhere
17       |   %3[0-9]: a string of at most 3 decimal digits (digits 5 and 6) => str2
18       |   %2c: two characters      => carr
19       |   ' ': all consecutive whitespace */
20
21    printf("Converted %d fields:\ni = %d\nx = %f\nstr1 = %s\n"
22          "j = %d\ny = %f\nstr2 = %s\n"
23          "carr[0] = U+%x\ncarr[1] = U+%x\n",
24          ret, i, x, str1, j, y, str2, carr[0], carr[1]);
25
26 } // Adapted from http://en.cppreference.com/w/c/io/fscanf
  
```

```

gcc version 4.6.3
>
Converted 7 fields:
i = 25
x = 5.432000
str1 = Thompson
j = 56
y = 789.000000
str2 = 56
carr[0] = U+61
carr[1] = U+62
>
  
```

Operadores (em ordem decrescente de prioridade)

Category	Operator	Associativity
Postfix	() [] -> . ++ --	Left to right
Unary	+ - ! ~ ++ -- (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR		Left to right
Logical AND	&&	Left to right
Logical OR		Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %= >>= <<= &= ^= =	Right to left
Comma	,	Left to right

Para usar outras funções matemáticas incluir as livrerias [math.h](#) e [stdlib.h](#). Ver mais informação [aqui](#).

Instruções Condicionais

```
if(boolean_expression) {  
    /* statement(s) will execute if the boolean expression is true */  
} else {  
    /* statement(s) will execute if the boolean expression is false */  
}
```

```
switch(expression) {  
  
    case constant-expression :  
        statement(s);  
        break; /* optional */  
  
    case constant-expression :  
        statement(s);  
        break; /* optional */  
  
    /* you can have any number of case statements */  
    default : /* Optional */  
        statement(s);  
}
```

Laços

```
while(condition) {
    statement(s);
}
```

```
do {
    statement(s);
} while( condition );
```

```
for ( init; condition; increment ) {
    statement(s);
}
```

```
1  #include <stdio.h>           // headers
2
3  int main(void){
4      int i, j, n;             // local variables definition
5      scanf("%d", &n);
6      for( i = 3; i < n; i+= 2 ){
7          j = 2;
8          while( j <= (i / j) ) { // how to avoid the division?
9              if( !( i % j ) ) break;
10             j++;
11         }
12         if( j > i / j ) printf("%d\n", i);
13     }
14     return 0;
15 }
```

O que faz esse programa?

Agenda

Introdução

Elementos básicos da linguagem C

Ambiente de trabalho

Vetores e Strings

Funções

Estudo independente

Exercícios para casa

Bibliografia

Exemplos de ambientes de trabalho para C

- ▶ Editor de texto + gcc na linha de comando, e.g. em Linux

```
$ gcc hello.c -o hello.exe  
$ ./hello.exe
```

- ▶ Ambiente de desenvolvimento integrado (IDE):
 - Netbeans para C/C++, [Tutorial de instalação Linux, Windows](#)
 - CodeBlocks, [Tutorial de instalação](#)
- ▶ Plataforma online
 - AWS Cloud9
<https://aws.amazon.com/pt/cloud9/?origin=c9io>
 - CodingGround
www.tutorialspoint.com/online_c_compiler.php
 - <https://repl.it>

Exercício 1 - Sequência de ADN

Uma sequência de ADN ou sequência genética é uma série de letras A, C, G e T, representando os quatro nucleotídeos de uma cadeia de ADN - as bases adenina, citosina, guanina, timina

Exemplo: ACAAGATGCCATTGTCCCCGGCCTCCTGCTGCTGC
TGCTCTCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGG e
GTGGCCCCACCGGCCGAGACAGCGAGCA são exemplos de
sequências de ADN

- ▶ Escreva um programa que leia uma sequência letras terminada no caractere '\n' e determine se corresponde ou não a uma sequência genética. Seu programa deve usar a menor quantidade de memória possível.

Exercício 2 - Sequência Complementar de ADN

Cada sequência de ADN está ligada a uma sequência complementar onde A liga-se com T e C com G. Como resultado desta complementariedade, toda a informação contida numa das cadeias de ADN está também contida na outra, o que é fundamental para a replicação do ADN.

Exemplo: AAAGTCTGAC e TTTCAGACTG são sequências complementares

- ▶ Escreva um programa que leia duas sequências de letras terminadas no caractere '`\n`', uma após a outra. Seu programa deve imprimir 0 se as duas sequências genéticas são complementares e 1 em outro caso, usando a menor quantidade de memória possível.

Exercício 2 - Sequência Complementar de ADN

Cada sequência de ADN está ligada a uma sequência complementar onde A liga-se com T e C com G. Como resultado desta complementariedade, toda a informação contida numa das cadeias de ADN está também contida na outra, o que é fundamental para a replicação do ADN.

Exemplo: AAAGTCTGAC e TTTCAGACTG são sequências complementares

- ▶ Escreva um programa que leia duas sequências de letras terminadas no caractere '`\n`', uma após a outra. Seu programa deve imprimir 0 se as duas sequências genéticas são complementares e 1 em outro caso, usando a menor quantidade de memória possível. **Assuma que o tamanho máximo das sequências é 100.**

Agenda

Introdução

Elementos básicos da linguagem C

Ambiente de trabalho

Vetores e Strings

Funções

Estudo independente

Exercícios para casa

Bibliografia

Arrays (Vetores ou Arranjos)

Tipo de dados que permite armazenar um número fixo de valores dum mesmo tipo base e que podem ser referenciados usando um único identificador de variável.

▶ Declaração: `TipoBase identificadorVar[NumElem]`;
`int v1[3]; int n = 3; float v2[n];`

▶ Declaração com Inicialização:

```
int v3[100]={0,1,2,3},  
float v4[2]={1.5,2,3}; // instrução não válida
```


▶ Após a criação, não é possível mudar o número de elementos

▶ Acesso aos elementos (índices `0..NumElem-1`):

```
v1[0]=1; v2[n-1]=v2[n-2];
```

```
//instruções válidas porém semanticamente erradas  
v1[-1]=65; v2[n]=12.3;
```

Arrays (Vetores ou Arranjos)

```
main.c   
1  #include <stdio.h>  
2  
3  void main() {  
4      int n;  
5      printf("Array size: "); scanf("%d", &n);  
6      int v[n];  
7      for(int i = 0; i < n; i++){  
8          printf("v[%d] = \n", i);  
9          scanf("%d", &v[i]);  
10         if ( v[i] & 1 )  
11             printf("\n%d -> B\n", v[i]);  
12         else  
13             printf("\n%d -> A\n", v[i]);  
14  
15         // Can we optimize this code?  
16     }  
17 }  
18
```

O que faz esse programa?

Outros tipos estruturados - Strings

Uma string é um array de caracteres terminados por um caractere nulo ou valor zero.

- ▶ Declaração: `char identificadorVar[NumElem];`
`char s1[5]; int n = 5; char s2[n]; char *s3;`
- ▶ Declaração com Inicialização:
`char s4[]={ 'H', 'o', 'l', 'a', 0 }, *s5 = "Olá";`
- ▶ Diferentemente dos vetores, as strings podem ser imprimidas e lidas diretamente
`printf("— %s!", s4);`
`scanf("%s", s1);`
`gets(s2);`
`fgets(s2, 4, stdin); // opção mais recomendada`

Agenda

Introdução

Elementos básicos da linguagem C

Ambiente de trabalho

Vetores e Strings

Funções

Estudo independente

Exercícios para casa

Bibliografia

returnType functionName(parameters) { inst }

- ▶ Somente podem devolver um valor de tipo básico ou ponteiro
- ▶ A instrução `return` pode estar em qualquer lugar do corpo
- ▶ A chamada `function_name(arguments)` deve ter correspondência com a declaração. A transferência entre os parâmetros e os argumentos da chamada é **sempre** por valor

```
1  #include<stdio.h>
2
3  void readIntArr( int n, int arr[] );
4
5  void writeIntArr( int n, int arr[] );
6
7  int main()
8  {
9      int numEl, n;
10     printf("Digite o número de elementos do vetor: ");
11     scanf("%d", &numEl);
12     int v[numEl];
13     readIntArr(numEl, v);
14     writeIntArr(numEl, v);
15     return 0;
16 }
```

Exercício 2 - Sequência Complementar de ADN

Cada sequência de ADN está ligada a uma sequência complementar onde A liga-se com T e C com G. Como resultado desta complementariedade, toda a informação contida numa das cadeias de ADN está também contida na outra, o que é fundamental para a replicação do ADN.

Exemplo: AAAGTCTGAC e TTTCAGACTG são sequências complementares

- ▶ Escreva um programa que leia duas sequências de letras terminadas no caractere '`\n`', uma após a outra. Seu programa deve imprimir 0 se as duas sequências genéticas são complementares e 1 em outro caso, usando a menor quantidade de memória possível. **Assuma que o tamanho máximo das sequências é 100.**



Exercício 3 - Sequência ARNm

O ARN mensageiro (ARNm) leva ao ribossoma a informação genética para a síntese de proteínas específicas. O ARNm utiliza a sequência das bases G, A, U, e C que significam guanina, adenina, uracilo e citosina. A síntese do ARNm usa uma sequência de ADN como modelo num processo conhecido como transcrição. O ARNm resultante da transcrição é uma sequência de ADN complementar onde a timina é substituída por uracilo.

Exemplo: A sequência UUU CAG ACU corresponde à transcrição da sequência AAA GTC TGA.

Cada conjunto de três bases consecutivas de ARNm (codão) é responsável pela codificação de um aminoácido ou indicam o ponto de início ou fim de tradução da cadeia de ARNm. A seguinte tabela mostra a relação entre os codões e os respectivos aminoácidos.

		Second base of codon								
		U		C		A		G		
First base of codon	U	UUU	Phenylalanine phe	UCU	Serine ser	UAU	Tyrosine tyr	UGU	Cysteine cys	U
		UUC		UCC		UAC		UGC		C
		UUA	Leucine leu	UCA		STOP codon	UAA	STOP codon	UGA	A
		UUG		UCG			UAG		UGG	Tryptophan trp
	C	CUU	Leucine leu	CCU	Proline pro	CAU	Histidine his	CGU	Arginine arg	U
		CUC		CCC		CAC		CGC		C
		CUA		CCA		CAA	Glutamine gin	CGA		A
		CUG		CCG		CAG		CGG		G
	A	AUU	Isoleucine ile	ACU	Threonine thr	AAU	Asparagine asn	AGU	Serine ser	U
		AUC		ACC		AAC		AGC		C
		AUA	ACA	AAA		Lysine lys	AGA	Arginine arg	A	
		AUG	Methionine met (start codon)	ACG			AAG		AGG	G
	G	GUU	Valine val	GCU	Alanine ala	GAU	Aspartic acid asp	GGU	Glycine gly	U
		GUC		GCC		GAC		GGC		C
		GUA		GCA		GAA	Glutamic acid glu	GGA		A
		GUG		GCG		GAG		GGG		G



Exercício 3 - Sequência ARNm

Exemplo: A sequência TACGGACATAACACCTGCATC gera o ARNm AUGCCUGUAUUGUGGACGUAG que gera a sequência de aminoácidos: **MET PRO VAL LEU CYS THR STOP**.

- ▶ Escreva um programa que leia uma sequência de letras terminada no caractere '\n' e determine a sequência de aminoácidos que será codificada. Essa sequência deve começar como o códon de começo (**MET** - que também pode aparecer em outras partes da sequência) e terminar com um códon **STOP**. Nesse caso, seu programa deve imprimir 0 seguido da sequência de aminoácidos codificados com 3 letras maiúsculas (exceto o **STOP**) separados por um espaço ou 1 em caso contrário. Use a menor quantidade de memória possível. **Assuma que o tamanho máximo da sequência de entrada é 1000.**

Agenda

Introdução

Elementos básicos da linguagem C

Ambiente de trabalho

Vetores e Strings

Funções

Estudo independente

Exercícios para casa

Bibliografia

Estruturas

Enquanto os vetores permitem armazenar, vários elementos de dados dum único tipo usando só um identificador de variável, as estruturas permitem armazenar elementos de dados de tipos diferentes.

Para acessar cada elemento de dado numa estrutura (*campo-field*) é usado o operador ponto (.)

```
struct [structure tag] {  
    member definition;  
    ...  
    member definition;  
} [one or more structure variables];
```

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

```
struct Books {  
    char title[50];  
    char author[50];  
    char subject[100];  
    int book_id;  
};
```

```
void printBook( struct Books book ) {  
    printf( "Book title : %s\n", book.title);  
    printf( "Book author : %s\n", book.author);  
    printf( "Book subject : %s\n", book.subject);  
    printf( "Book book_id : %d\n", book.book_id);  
}
```

Números (pseudo-) aleatórios

Para gerar números aleatórios deve ser usada a função `rand()`. No entanto, antes deve ser usada a função `srand()`, uma vez no início do programa. Ver mais informações [aqui](#).

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

int main(void)
{
    srand(time(NULL)); //use current time as seed for random generator
    int random_variable = rand();
    printf("Random value on [0,%d]: %d\n", RAND_MAX, random_variable);
}
```

Possible output:

```
Random value on [0 2147483647]: 1373858591
```


Agenda

Introdução

Elementos básicos da linguagem C

Ambiente de trabalho

Vetores e Strings

Funções

Estudo independente

Exercícios para casa

Bibliografia



Exercícios para casa

Escreva programas C (definindo funções apropriadas) para:

1. dado um inteiro $k > 1$, gerar uma sequência de M números inteiros aleatórios e determinar se a sequência contém pelo menos uma sub-sequência de até k números consecutivos. Faça o programa imprimir todas as sub-sequências máximas

Exemplo: Para $k = 4$ suponha que a sequência aleatória é 8,-10,4,-2,-1,0,1,2,50,51,54. Essa sequência contém as sub-sequências -2,-1,0,1; -1,0,1,2 e 50,51

2. gerenciar uma agenda de contatos. Para cada contato deve-se armazenar **Nome**, **Telefone** e **Email**. Seu programa deve dar as seguintes opções ao usuário: **Inserir um contato** no final da agenda, **Buscar dados pelo Nome**, **Buscar dados pelo Telefone**, **Mostrar Agenda** e **Sair**.

Agenda

Introdução

Elementos básicos da linguagem C

Ambiente de trabalho

Vetores e Strings

Funções

Estudo independente

Exercícios para casa

Bibliografia

Bibliografia e Links úteis

- ▶ **Beginning C**, Ivor Horton, 5th ed. 2013
<https://github.com/apress/beg-c-5th-edition>
- ▶ **C How to Program**, Paul J. Deitel & Harvey Deitel, 8th ed. 2015
- ▶ **C Programming Language**, Brian W. Kernighan & Dennis Ritchie. 1988
- ▶ **Essential C**, Nick Parlante. 2003
<http://cslibrary.stanford.edu/101/EssentialC.pdf>
- ▶ <https://www.tutorialspoint.com/cprogramming/index.htm>
- ▶ **Slides de Programação Estruturada**,
Fabrício Olivetti de França, Jesús P. Mena-Chalco, Paulo Henrique Pisani