C Referência Rápida

ESTRUTURA DO PROGRAMA

tipo fnc(*tipo1*,…)	function declarations
tipo nome;	declaração de variáveis externas
main() {	função principal
declarações;	variáveis locais
comandos; }	
tipo fnc (*arg1*,…) {	definição de função
declarações;	variáveis locais
comandos;	corpo da função
return *valor*; }	valor de retorno
/* */	comentários

PRÉ-PROCESSADOR

Inclui biblioteca bib	**#include <***bib***>**
Inclui arquivo bib	**#include "***bib***"**
Texto de troca	**#define** *nome texto*
Macro de troca	**#define** *nome* **(var)** *texto*
Apaga a definição	**#undef** *nome*
Continuação de linha	\

TIPOS DE DADOS / DECLARAÇÕES

caractere (1 byte)	**char**
inteiro	**int**
flutuante (single precision)	**float**
flutuante (double precision)	**double**
short (16 bit integer)	**short**
long (32 bit integer)	**long**
positive e negativo	**signed**
somente positivo	**unsigned**
ponteiros para int, float	***int, *float,**
enumeração	**enum**
valor constante	**const**
variável externa	**extern**
variável registro	**register**
local ao código	**static**
sem valor	**void**
estrutura	struct
cria um novo tipo	**typedef** *nomedotipo*

INICIALIZAÇÃO

Inicializa qualquer tipo	*tipo nome = valor;*
Inicializa vetores	*vetor* [ ]={*val1*, *val2*, ...};
Inicializa strings	char *nome* [ ]="*string*";

FLUXO DE CONTROLE

Final de comando	;
Delimitador de blocos	{ }
Saída de switch, for, do, while	**break**
Próxima interação for, do, while	**continue**
Retorno de valor de função	**return** expressão;

```
if (condição)      comando;
else               comando;
if (condição)    {
      comando1;
      comando2;  }
else {
      comando3;
      comando4;  }
switch (expr) {
  case const1: comando; break;
  case const2: comando; break;
  default: comando; }
```

ESTRUTURAS REPETIÇÃO

```
for (i = valor; condição; incremento)
  comando;
while (condição)
  comando;
do {  comando;  → sempre entre { }
  } while (condição);
```

VETORES

```
int vetor[10], cont, matriz[10][3];
for (cont=0; cont<10; cont++) {
  printf ("\nDigite um valor inteiro: ");
  scanf ("%d", &vetor[cont]);
}
for (cont=0; cont<10; cont++)
printf ("%d = %d\n", cont, vetor[cont]);
```

Palavras reservadas da linguagem C

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

PONTEIROS

Declara ponteiro ao tipo	*tipo * nome*
Ponteiro nulo	**NULL**
Objeto apontado por ponteiro	**ponteiro*
Endereço do objeto nome	*&nome*

EXEMPLO:

int x, *y;	declara variável X e ponteiro Y
x = *y;	**ERRO**: Y não inicializado!!!
x = 10;	inicializa X
y = &x;	inicializa ponteiro com endereço de x
printf("%d", x);	imprime o valor de x
printf("%d", y);	LIXO
printf("%d", *y);	imprime o valor apontado por y
printf("%p", y);	imprime o endereço contido em y

FUNÇÕES

Declarando funções:

(se uma função possuir um tipo de retorno diferente de "void" é necessário retornar um valor utilizando o comando "return")

tipoRetorno nomeFunção (parâmetros)
{ comandos; → SEMPRE entre { }
}

Chamando uma função:

int x = soma(2,2);

Passagem de parâmetro por valor:

(a variável declarada na lista de parâmetros recebe o VALOR da variável utilizada na chamada da função)

tipoRetorno nomeFunção (tipo par, tipo par, ...)

Passagem de parâmetro por referência:

(alterando a variável do parâmetro, altera também a variável utilizada na chamada da função)

(um '*' é adicionado no final do tipo para representar que o parâmetro é uma referência)

tipoRetorno nomeFunção (tipo par, tipo* par, ...)*

ESTRUTURAS

struct nome {	nome da estrutura
declarações;	membros
};	
Cria estrutura	**struct** *tag nome*
Membro da estrutura	*nome.membro*
Membro apontado pela estrutura	*pointer -> membro*

OPERADORES (POR PRECEDÊNCIA)

Operador membro da estrutura	*nome. membro*
Ponteiro da estrutura	*ponteiro-> membro*
Incremento, decremento	++, --
Mais, menos, não, não bit	+, -, !, ~
Valor apontado, endereço do ponteiro	* *ponteiro*, & *nome*
Mudando o tipo da expressão	(*tipo*) *expressão*
Tamanho de objeto	**sizeof**
Multiplica, divide, módulo (resto)	*, /, %
Adiciona, subtrai	+, -
Shift a esquerda, a direito [bit]	<<, >>
Comparações	>, >=, <, <=
Comparações	==, !=
AND bits	&
OR exclusivo bits	^
OR bits	|
AND lógico	&&
OR lógico	||
Operadores de atribuição	+=, -=, *=, : : :
Separador de expressões	,

BIBLIOTECAS PADRÃO

<assert.h>	<ctype.h>	<errno.h>	<float.h>	<limits.h>
<locale.h>	<math.h>	<setjmp.h>	<signal.h>	<stdarg.h>
<stddef.h>	<stdio.h>	<stdlib.h>	<string.h>	<time.h>

STRING

S e T são strings, CS e CT são constants

Tamanho de S	**strlen**(s)
Copia CT para S	**strcpy**(s,ct)
Copia até N caracteres	**strncpy**(s,ct,n)
Concatena CT após S	**strcat**(s,ct)
Concatena até N caracteres	**strncat**(s,ct,n)
Compara CS a CT	**strcmp**(cs,ct)
Compara somente N caracteres	**strncmp**(cs,ct,n)
Ponteiro para o primeiro C em CS	**strchr**(cs,c)
Ponteiro para o último C em CS	**strrchr**(cs,c)
Copia N chars de CT para S	**memcpy**(s,ct,n)
Copia N chars de CT para S (sobrepor)	**memmove**(s,ct,n)
Compara N chars de CS com CT	**memcmp**(cs,ct,n)
Ponteiro para o primeiro C nos primeiros N chars de CS	**memchr**(cs,c,n)
Coloca C no primero N chars de CS	**memset**(s,c,n)

STDIO.H, CSTDIO (entrada/saída)

FILE* f=**fopen**("filename", "r");	Abre para leitura
	Outros modos: w, a, a+, rb
fclose(f);	Fecha arquivo F
fprintf(f, "x=%d", 3);	imprime "x=3"
outros formatos:	
"%5d %u %-8ld"	5 casas, unsigned, long à esquerda
"%o %x %X %lx"	octal, hexa, HEXA, long hex
"%f %5.1f"	float or double: 123.000000, 123.0
"%e %g"	1.23e2, use either f or g
"%c %s"	char, char*
"%%"	%
sprintf(s, "x=%d", 3);	Imprime para vetor de chars S
printf("x=%d", 3);	Imprime para tela
fprintf(stderr, ...	Imprime para erro padrão
getc(f);	Lê um char
ungetc(c, f);	Devolve um C para arquivo F
getchar();	**getc**(stdin);
putc(c, f)	**fprintf**(f, "%c", c);
putchar(c);	**putc**(c, stdout);
fgets(s, n, f);	Lê linha para S de arquivo
gets(s)	**fgets**(s, INT_MAX, f);
fread(s, n, 1, f);	Lê N butes de F para S
fwrite(s, n, 1, f);	Escreve N bytes de S para F
fflush(f);	Força buffer
fseek(f, n, SEEK_SET);	Posiciona arq. binário F em N
ftell(f);	Posição em F, -1L se erro
rewind(f);	**fseek**(f, 0L, SEEK_SET);
clearerr(f);	Limpa erro
feof(f);	Arq F no final?
ferror(f);	Arq F com erro?
remove("filename");	Apaga arquivo, 0 se OK
rename("old", "new");	Renomeia arquivo, 0 se OK
f = tmpfile();	Cria arq temp em modo wb+

STDLIB

Valor absoluto	**abs**(n)
Valor absolute longo	**labs**(n)
Quociente e resto	**div**(n,d)
Quociente e resto, longo	**ldiv**(n,d)
Pseudo-aleatório	[0,RAND_MAX] **rand**()
Semente aleatória	**srand**(n)
Termina execução	**exit**(status)
Execução do sistema	**system**(s)

Conversions

string s → double	**atof**(s)
string s → integer	**atoi**(s)
string s → long	**atol**(s)

TIME

Processador de tempo	**clock**()
Tempo calendário corrente	**time**()
Diferença em segundos	**ifftime**(time2,time1)
Tipos aritméticos para tempo	**clock_t, time_t**

Estrutura para definer components de tempo **tm**

tm_sec	segundos depois de minutos
tm_min	minutes depois de hora
tm_hour	horas desde a meia-noite
tm_mday	dia do mês
tm_mon	mês desde janeiro
tm_year	anos desde 1900
tm_wday	dias desde domingo
tm_yday	dias desde 1/janeiro
tm_isdst	flag para horário de verão

tempo local → tempo calendário	**mktime**(tp)
tempo → string	**asctime**(tp)
tempo calendário → tempo local	**ctime**(tp)
tempo calendário → GMT	**gmtime**(tp)
tempo calendário → tempo local	**localtime**(tp)