Cartão de Referência do C (ANSI)

Estrutura do Programa / Funções

tipo fun(type1,...);	protótipo da função
tipo nome;	declaração de variável
int main(void) {	rotina principal
declarações	declaração de variáveis locais
comandos	
return *valor;*	
}	
/* */	comentários
int main(int argc, char *argv[])	principal com argumentos
exit(*arg*);	término de execução

Preprocessador C

inclusão de arquivos de cabeçalho	#include *<nomearquivo>*
inclusão de arquivos do usuário	#include "*nomearquivo*"
substituição de texto	#define *nome texto*
substituição de macro	#define *nome(var) texto*
Exemplo: #define *max(A,B) ((A)>(B) ? (A) : (B)*	
anular definição	#undef *nome*
cotação ao substituir	#
Exemplo: #define msg*(A)* printf*("%s = %d, #A, (A))*	
concatena argumentos e substitui	##
compilação condicional	#if, #else, #elif, #endif
nome definido?, não definido?	#ifdef, #ifndef
nome definido	defined(nome)
caractere de continuação de linha	\

Tipo de dados / Declarações

caractere (1 byte)	char
inteiro	int
real (simples, dupla precisão)	float, double
curto (inteiro de 16 bits)	short
longo (inteiro de 32 bits)	long
longo duplo (inteiro de 32 bits)	long long
positivo ou negativo	signed
somente positivo	unsigned
ponteiro para int, float	int*, float*
enumeração	enum *tag {nome1=valor1, …}*;
valor constante(só leitura)	*type* const *nome;*
declaração de variável externa	extern
variável em registrador	register
interna ao arquivo fonte	static
local persistente entre chamadas	static
nenhum valor	void
estrutura	struct tag {...}
criar ou renomear um tipo de dado	typedef *tipo nome;*
tamanho de um obj. (tipo é size_t)	sizeof *object*
tam. de um tipo de dado (tipo é size_t)	sizeof(*dado*)

Inicialização

inicializa variável	*tipo nome=valor;*
inicializa vetor	*tipo nome*[]={*valor1,...*};
inicializa string (cadeia de caracteres)	char *nome*[]="*string*";

Constantes

sufixos: long, unsigned float	3.0L, -1U, 3.0F
forma exponencial	4.2e1
prefixos: octal, hexadecimal	0, 0x ou 0X
Exemplo: 031 é 25 e 0x31 é 49 decimal	
constante caractere (char, octal,hex)	'a', '\ooo','xhh'
nova linha, cr, tab, backspace	\n, \r, \t, \b
caracteres especiais	\\,\?, \', \"
constante string (termina com '\0')	"abc...de"

Ponteiros, Vetores e Estruturas

declara ponteiro para *tipo*	*tipo *name*
declara função retornando ponteiro para *tipo*	*tipo* *f()
declara ponteiro para função retornando *tipo*	*tipo* (*pf)()
ponteiro genérico	void *
ponteiro constante nulo	NULL
objeto apontado por ponteiro	**ponteiro*
endereço do objeto nome	&*nome*
vetor	*nome[dim]*
vetor multidimensional	*nome[dim$_1$][dim$_2$]...*

Estruturas

struct *etiqueta* {	modelo de estrutura
declarações	declaração de membros
};	
criar estrutura	struct *etiqueta nome*
membro da estrutura	*nome.membro*
membro da estrutura apontado por	nome -> membro
Exemplo: (*p).x e p -> x são o mesmo	
objeto único, múltiplos tipos possíveis	union
Campo de bits com b bits	unsigned *campo*:b;

Operadores (agrupados por precedência)

operador membro da estrutura	*nome.membro*
membro da estrutura através de ponteiro	*ponteiro->membro*
incremento, decremento	++, --
mais, menos,não lógico, não bit a bit	+, -, !, ~
indireção por ponteiro, endereço do objeto	**ponteiro*, &*nome*
converte expressão para tipo	(*tipo*)expr
tamanho de um objeto	sizeof
multiplicação, divisão, módulo (resto)	*, /, %
adição, subtração	+, -
deslocamento à esquerda, direita (bit a bit)	<<, >>
operadores relacionais	>, >=, <, <=
operadores relacionais	==, !=
e (bit a bit)	&
ou exclusivo (bit a bit)	^
ou (bit a bit)	|
e lógico	&&
ou lógico	||
expressão condicional	expr$_1$? expr$_2$: expr$_3$
operadores de atribuição	=, +=, -=,*=, ...
separador de avaliação de expressões	,

Operadores unários, expressões condicionais e operadores de atribuição são agrupados da direita para à esquerda; todos os outros grupos da esquerda para a direita.

Controle de Fluxo

finalizador de comando	;
delimitadores de bloco	{ }
saída de switch, while, do, for	break;
próxima iteração de while, do, for	continue;
ir para	goto *rótulo*;
rótulo	rótulo: *comando*;
retorna valor de função	return *expr*;

Construções de Fluxo

comando if	if (*expr$_1$*) *comando$_1$;*
	else if (*expr$_2$*) *comando$_2$*;
	else *comando$_3$*;
comando while	while (*expr*)
	comando;
comando for	for (*expr$_1$; expr$_2$; expr$_3$*)
	comando;
comando do	do *comando*;
	while (*expr*);
comando switch	switch (*expr*) {
	case *const$_1$*: *comando$_1$*; break;
	case *const$_2$*: *comando$_2$*; break;
	default: *comando*;
	}

Bibliotecas ANSI 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>

Consulta de tipo caractere <ctype.h>

c é um caractere

alfanumérico?	isalnum(c)
alfabético?	isalpha(c)
caractere de controle?	iscntrl(c)
dígito decimal?	isdigit(c)
caractere imprimível? (excluído o espaço)	isgraph(c)
letra minúscula?	islower(c)
caractere imprimível? (incluído o espaço)	isprint(c)
char imprimível exceto espaço, letra, dígito?	ispunct(c)
espaço, formfeed, newline, cr, tab, vtab?	isspace(c)
letra maiúscula?	isupper(c)
dígito hexadecimal?	isxdigit(c)
converter para minúscula	tolower(c)
converter para maiúscula	toupper(c)

Operações com strings <string.h>

s é uma string; cs e ct são são strings constantes

comprimento de s	strlen(s)
copiar ct em s	strcpy(s,ct)
copiar ct em s até n caracteres	strncpy(s,ct,n)
concatenar ct após s	strcat(s,ct)
concatenar ct após s até n caracteres	strncat(s,ct,n)
comparar cs com ct	strcmp(cs,ct)
… somente os primeiros 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)
copiar n caracteres de ct em s	memcpy(s,ct,n)
copiar n chars. de ct em s (sobrescrever)	memmove(s,ct,n)
comparar n caracteres de cs com ct	memcmp(cs,ct,n)
ponteiro ao 1º c nos n 1os chars de cs	memchr(cs,c,n)
por c nos n primeiros caracteres de cs	memset(s,c,n)

Cartão de Referência do C (ANSI)

Entrada/Saída <stdio.h>

E/S Padrão

fluxo de entrada padrão	stdin
fluxo de saída padrão	stdout
fluxo de erro padrão	stderr
fim de arquivo (tipo é int)	EOF
obter um caractere	getchar()
imprimir um caractere	putchar(*char*)
imprimir dados formatados	printf(*"formato",arg1,...*)
imprimir em uma string s	sprintf(*s,"formato",arg1,...*)
ler dados formatados	scanf(*"format",&nome1,...*)
ler de uma string	sscanf(*s,"format",&nome1,...*)
obter string s	gets(*s*)
imprimir string s	puts(*s*)

E/S em Arquivo

declara ponteiro para arquivo	FILE *fp
obter ponteiro para arquivo	fopen(*"nome","modo"*)
Modos: r (leitura), w (escrita), a (anexo), b (binário)	
obter um caractere	getc()
escrever um caractere	putc(*char*)
escrever em arquivo	printf(*fp,"formato",arg1,...*)
ler de arquivo	fscanf(*fp,"formato",&nome1,...*)
ler e guardar n dados em *ptr	fread(*ptr,tamanho_dado,n,fp*)
escrever n dados de *ptr no arq	fwrite(*ptr,tamanho_dado,n,fp*)
fechar arquivo	fclose(*fp*)
não zero se erro	ferror(*fp*)
não zero se alcançou EOF	feof(*fp*)
ler linha para string s (< *max cars*)	fgets(*s,max,fp*)
escrever string em arquivo	fputs(*s,fp*)

Códigos para E/S formatada: *"%-+ 0w.pmc"*

-	justificado à esquerda
+	imprime com sinal
espaço	imprime espaço se não há sinal
0	preencher na frente com zeros
w	largura mínima do campo
m	caractere de conversão: h: short, l: long, L: long double
c	caractere de conversão:
d,i: inteiro	u: sem sinal
c: caractere	s: string
f: double (printf)	e,E: exponencial
f: float (scanf)	lf: double (scanf)
o: octal	x,X: hexadecimal
p: ponteiro	n: número de chars escritos
g,G: mesmo que f ou e,E dependendo do expoente	

Lista Variável de Argumentos <stdarg.h>

declara ponteiro para argumentos	va_list *ap*;
inicializar ponteiro à argumentos	va_start(*ap,ultimo_arg*)
ultimo_arg é o último parâmetro com nome da função	
seguinte arg. sem nome, atualiza ponteiro	va_arg(*ap,tipo*)
chama antes de sair da função	va_end(*ap*)

Funções Úteis <stdlib.h>

Valor absoluto do inteiro n	abs(n)
Valor absoluto do longo n	labs(n)
quociente e resto de inteiros n,d	div(n,d)
devolve uma estrutura com div_t.quot e div_t.rem	
quociente e resto de longos n,d	ldiv(n,d)
devolve uma estrutura com ldiv_t.quot e ldiv_t.rem	
inteiro pseudoaleatório em [0,RAND_MAX]	rand()
fixar uma semente aleatória a n	srand()
finalizar a execução do programa	exit(estado)
executar string s no sistema	system(s)

Conversões

converter string s em double	atof(s)
converter string s em inteiro	atoi(s)
converter string s em longo	atol(s)
converter prefixo de s em double	strtod(s,&endptr)
converter prefixo de s (base b) em long	strtol(s,&endptr,b)
igual, porém unsigned long	strtoul(s,&endptr,b)

Reserva de Memória

reserva memória	malloc(dim), calloc(nobj,dim)
trocar tamanho da reserva	novoptr = realloc(ptr,dim)
liberar memória	free(ptr)

Funções de Vetores

buscar chave no vetor	bsearch(chave,vetor,n,dim,cmpfun)
ordenar vetor ascendentemente	qsort(vetor,n,dim,cmpfun)

Funções de hora e data <time.h>

tempo de processamento usado pelo programa	clock()
Exemplo: clock()/CLOCKS_PER_SEC da o tempo em segundos	
segundos desde 1/1/1970 (hora de ref.)	time()
tpo_2-tpo_1 em segundos (double)	difftime(tpo_2,tpo_1)
tipos numéricos para representar horas	clock_t,time_t
estrutura padrão usada para data e hora	struct tm

tm_sec	segundos após minuto
tm_min	minutos após hora
tm_hour	horas desde a meia-noite
tm_mday	dia do mês
tm_mon	meses desde janeiro
tm_year	anos desde 1900
tm_wday	dias desde domingo
tm_yday	dias desde 1º de janeiro
tm_isdst	indicador de horário de verão

converter hora local na hora de ref.	mktime(tp)
converter hora em tp na string	asctime(tp)
converter hora de ref. em tp na string	ctime(tp)
converter hora de ref. em GMT	gmtime(tp)
converter hora de ref. na hora local	localtime(tp)
formatar data e hora	strftime(s,smax,*"formato"*,tp)

tp é um ponteiro para uma estrutura de tipo tm

Funções Matemática <math.h>

Os argumentos e os valores devolvidos são double

funções trigonométricas	sin(x), cos(x), tan(x)
funções trigonométricas inversas	asin(x), acos(x), atan(x)
arctan(y/x)	atan2(y,x)
funções trig. hiperbólicas	sinh(x), cosh(x), tanh(x)
exponenciais e logaritmos	exp(x), log(x), log10(x)
exponenciais e logaritmos base 2	ldexp(x,n), frexp(x,*e)
divisão e resto	modf(x,*ip), fmod(x,y)
potência e raiz quadrada	pow(x,y), sqrt(x)
arredondamento	ceil(x), floor(x), fabs(x)

Limites do tipo inteiro <limits.h>

Os números em parênteses são valores típicos para as constantes em um sistema UNIX de 32 bits, seguidos pelos valores mínimos requeridos (se significativamente diferente.

CHAR_BIT	bits em char	(8)
CHAR_MAX	máximo valor de char	(SCHAR_MAX ou UCHAR_MAX)
CHAR_MIN	mínimo valor de char	(SCHAR MIN or 0)
SCHAR_MAX	máximo signed char	(+127)
SCHAR_MIN	mínimo signed char	(-128)
SHRT_MAX	máximo valor de short	(+32.767)
SHRT_MIN	mínimo valor de short	(-32.768)
INT_MAX	máximo valor de int	(+2.147.483.647))
INT_MIN	mínimo valor de int	(-2.147.483.647)
LONG_MAX	máximo valor de long	(+2.147.483.647)
LONG_MIN	mínimo valor de long	(-2.147.483.648)
UCHAR_MAX	máximo unsigned char	(255)
USHRT_MAX	máximo unsigned short	(65.535)
UINT_MAX	máximo unsigned int	(4.294.967.295)
ULONG_MAX	máximo unsigned long	(4.294.967.295)

Limites do tipo real <float.h>

FLT_RADIX	base usada para a representação	(2)
FLT_ROUNDS	modo de arredondamento	
FLT_DIG	dígitos decimais de precisão	(6)
FLT_EPSILON	menor x tal que 1,0 + x ≠ 1,0	($1{,}1 \times 10^{-7}$)
FLT_MANT_DIG	número de dígitos da mantissa	
FLT_MAX	máximo número float	($3{,}4 \times 10^{38}$)
FLT_MAX_EXP	exponente máximo	
FLT_MIN	minimo número float	($1{,}2 \times 10^{-38}$)
FLT_MIN_EXP	mínimo exponente	
DBL_DIG	dígitos decimais de precisão	(15)
DBL_EPSILON	menor x tal que 1,0 + x ≠ 1,0	($2{,}2 \times 10^{-16}$)
DBL_MANT_DIG	número de dígitos da mantissa	
DBL_MAX	máximo número double	($1{,}8 \times 10^{308}$)
DBL_MAX_EXP	exponente máximo	
DBL_MIN	minimo número double	($2{,}2 \times 10^{-308}$)
DBL_MIN_EXP	mínimo exponente	

Envie comentários e sugestões para J.H. Silverman, Math. Dept., Brown Univ., Providence, RI 02912 USA. (jhs@math.brown.edu)

Traduzido para o português por Marco Valério Miorim Villaça, Depto de Eletrônica, IF de Santa Catarina, Campus Florianópolis, Brasil.