C Reference Card (C99 / JIS X 3010)

プログラム構造と関数

関数宣言	`type func(type arg1, ...);`
変数宣言	`type var;`

main 関数	`int main(int argc, char *argv[]) {`
	`  制御文;`
	`}`

関数定義	`type func(arg1, ...) {`
	`  制御文;`
	`  return expr;`
	`}`

終了	`exit(arg);`
コメント	`/* ... (複数行可) */`
1 行コメント	`// ... (改行までコメント)`

構文・制御

命令終了文字	`;`
ブロック定義	`{ }`
if 選択文	`if (expr) { func_true(); }` `  else { func_false(); }`

switch 選択文	`switch (expr) {`
	`  case 定数1: func1(); break;`
	`  case 定数2: func2(); break;`
	`  default: func_defalut();`
	`}`

while 繰返し文	`while (expr) func();`
for 繰返し文	`for ( expr ; expr ; expr ) func();`
do 繰返し文	`do func(); while(expr);`
goto 分岐文	`goto label;`
goto 分岐文ラベル	`label:`
次実行分岐文 (while,do,for)	`continue;`
終了分岐文 (switch,while,do,for)	`break;`
関数終了分岐文	`return expr;`

C プリプロセッサ

ライブラリ読み込み	`#include <libraryName>`
ソースファイル読み込み	`#include "filename"`
マクロ置き換え	`#define name text`
マクロ関数定義	`#define name(var) text`
マクロ定義解除	`#undef name`
条件分岐	`#if, #else, #elif, #endif`
定義条件分岐	`#ifdef, #ifndef`
行制御・エラー・プログマ指令	`#line, #error, #pragma`
マクロ置き換え演算子	`#, ##`
空指令(行頭・コメントと同様)	`#`

数値・文字表現

8 進数表記	0	例. 071 (=57)
16 進数表記	0x, 0X	例. 0x39 (=57)
指数表記	e	例. 10e5 ($=10^5$)
文字	'a', '¥ooo', '¥xhh'	
文字列(終端'¥0')	"string"	
マルチバイト文字列(終端'¥0')	L"マルチバイト"	
newline/cr/tab/backspace	¥n, ¥r, ¥t, ¥b	
特殊文字 (¥,?,',")	¥¥, ¥?, ¥', ¥"	

データ型/宣言

型種類	宣言	接尾語
文字型(1Byte)	`char`	
ワイド文字型	`wchar_t`	
整数型	`int`	
short(16 bit 整数型)	`short`	
long (32 bit 整数型)	`long`	L
Long long(64 bit 整数型)	`long long`	LL
浮動小数点(単精度)	`float`	F
浮動小数点(倍精度)	`double`	
浮動小数点(倍精度)	`long double`	L
値無し	`void`	
複素数型	`double _Complex`	
虚数型	`double _Imaginary`	
Bool 型	`_Bool`	

列挙体・派生型

列挙体	`enum tag {`
	`  type membername;`
	`} var;`

構造体	`struct tag {`
	`  type membername;`
	`} var;`

共用体	`union tag {`
	`  type membername;`
	`} var;`

ポインタ型	`int *ptr; float *ptr;`
配列型	`int var[10]; float var[10];`

型修飾子・記憶クラス指定子

符号付き	`signed`	
符号無し	`unsigned`	接尾語「U」
定数修飾子 (値変更無し)	`const`	
restrict 修飾子	`restrict`	
最適化抑制修飾子	`volatile`	
外部宣言指定子	`extern`	
自動変数	`auto`	
静的変数	`static`	
レジスタ変数	`register`	
変数型定義指定子	`typedef type typename;`	
インライン関数指定子	`inline`	

初期化

変数	`type var = value;`
配列値	`type var[] = {val1, val2,...};`
文字列	`char var[] = "string";`

ポインタ・配列・構造体

ポインタ変数宣言	`type *ptr;`
ポインタ関数宣言	`type *func();`
アドレス参照代入	`ptr = &var;`
汎用ポインタ	`void *`
配列 (1 次元)	`type var[size]; // var[5]`
配列 (多次元)	`type var[size][size]...;`
可変配列定義	`type var[n]; // n は変数`
構造体変数宣言	`struct tag var;`
メンバ参照	`var.member;`
ポインタ参照	`ptr->member; // (*ptr).member`
ビットフィールド	`member : b // b はビット数`

演算子 (優先度順)

関数呼び出し	`func()`
配列添字	`var[]`
構造体メンバ参照・ポインタ参照	`var.member , ptr->member`
後置インクリメント・デクリメント	`var++ , var--`
前置インクリメント・デクリメント	`++var , --var`
アドレス演算子・参照演算子	`&var , *ptr`
単項 正数・負数・補数・論理否定	`+ , - , ~ , !`
型変換(キャスト)	`(type) expr`
sizeof 演算子	`sizeof`
乗算・除算・剰余	`* , / , %`
加算・減算	`+ , -`
左ビットシフト・右ビットシフト	`<< , >>`
比較演算子(大小比較)	`< , > , <= , >=`
比較演算子(等価・否定)	`== , !=`
論理積ビット演算(AND)	`&`
排他的論理和ビット演算(XOR)	`^`
論理和ビット演算(OR)	`|`
論理 AND	`&&`
論理 OR	`||`
条件三項演算子(?:)	`expr ? true() : false();`
代入	`=`
代入演算(加・減・乗・除・剰算)	`+= , -= , *= , /= , %=`
代入ビット演算 (左シフト・右シフト・AND・OR・XOR)	`<<= , >>= , &= , |= , ^=`
コンマ演算子	`,`

標準ライブラリ

<assert.h>	<errno.h>	<signal.h>	<stddef.h>
<stdio.h>	<stdlib.h>	<stdarg.h>	<setjmp.h>
<ctype.h>	<string.h>	<wchar.h>	<wctype.h>
<math.h>	<complex.h>	<tgmath.h>	<time.h>
<float.h>	<fenv.h>	<stdint.h>	<inttypes.h>
<limits.h>	<stdbool.h>	<locale.h>	<iso646.h>

予約後一覧

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

Tips

1 次元配列の要素数 `sizeof( array ) / sizeof( array[0] );`

1 次元配列初期化 `int array[2] = {0,1};`

1 次元配列初期化2 `int array[] = {0,1,3,4};`

2 次元配列初期化 `int array[][2] = { {0,1}, {2,3}, {4,5} };`

値入れ替え `#define SWAP(x,y) ((x!=y)?(x+=y,y=x-y,x-=y):0)`

C Reference Card (C99 / JIS X 3010)

入出力 <stdio.h>

標準入力ストリーム	stdin
標準出力ストリーム	stdout
標準エラーストリーム	stderr
ファイル終端(End Of File)	EOF
ファイルストリーム	FILE * fp

ファイル操作

ファイルオープン	fp = fopen("filename", "mode");
	mode: r(読込), w(書込), a(追記), +(更新), b(バイナリ)
ファイルクローズ	fclose(fp)
ファイルフラッシュ	fflash(fp)
EOF 検出(非ゼロ時 EOF)	feof(fp)
エラー検出(非ゼロ時エラー)	ferror(fp)
エラークリア	clearerr(fp)

文字・文字列の入出力

1 文字標準出力, fp へ出力	putchar(char), putc(c, fp)
1 文字標準読込, fp から読込	var = getchar(); getc(*fp*);
1 文字 fp へ押戻し	ungetc(char, *fp*)
文字列標準出力, fp へ出力	puts(*string*), fputs(str, *fp*)
文字列標準読込, fp から読込	gets(*str*),fgets(str, max, *fp*)
書式付き標準出力	printf(*format, arg)
書式付き fp へ出力	fprintf(*fp, *format, arg)
書式付き標準入力	scanf(*format, &arg)
書式付き fp から入力	fscanf(*fp, *format, &arg)

文字・文字列の実引数への書式入出力

書式付き出力	sprintf(str, *format, arg)
書式付き出力(文字数指定)	snprintf(str, n,*format, arg)
書式付き入力	sscanf(str, *format, &arg)

書式指定子 (書式指定文字列 %fw.plc)

	printf 系	scanf 系
フラグ(f)	- (左詰め), + (+符号表示), 空白(空白埋め), 0 (0 埋め), # (代替形式 o:01,x:0x1 等)	*代入抑止
フィールド幅(w)	最小幅	最大幅
精度(p)	有効桁数を数字表記	なし
長さ修飾子(l)	hh (diouxn) char 型 h (diouxn) short int 型 l (dioux) long int 型 (c) wint_t 型 (s) wwchar_t 型 (aefg) double 型 ll (diouxn) long long int 型 j (diouxn) intmax_t 型 or uintmax_t 型 z (diouxn) size_t 型 t (diouxn) ptrdiff_t 型 L (aefg) long double 型	
変数指定子(c)	d,i(10 進整数), o(8 進整数), x(16 進整数), u(符号無整数), f(小数 double), e(指数), g(f/e 自動選択),a(小数 16 進), c(1 文字), s(文字列), p(ポインタ), %(%文字)	f :float lf:double printf 系と違うため注意

一般ユーティリティ <stdlib.h>

整数算術

絶対値 int, long, long long 型	abs(), labs(), llabs()
商剰余 div_t, ldiv_t, lldiv_t 型	div(), ldiv(), lldiv()

探索・ソート

整列済み配列からの探索

```
bsearch(*key,*base,nmemb,size,int (*cmp)(void *,void *))
```

ソート

```
qsort(*base, nmemb, size, int (*cmp)(void *, void *))
```

擬似乱数

擬似乱数生成[0, RAND_MAX]	rand()
乱数種の設定	srand(n)

文字列→数値変換

Ascii to float, int, long, long long

atof(*string*), atoi(*str*), atol(*str*), atoll(*str*)

文字列→数値変換 (エラー検出付き)

String to float, double, long double

strtof(string, error_ptr), strtod(s,e), strtold(s,e)

String to long, long long, unsigned long, unsigned long long

strtol(s,e), strtoll(s,e), strtoul(s,e), strtoull(s,e)

動的メモリ管理

メモリ確保,初期化済	prt = malloc(size);calloc(nmemb,size);
メモリサイズ変更	realloc(ptr, size)
メモリ解放	free(ptr)

文字操作 <ctype.h>

文字判定 (一致時否ゼロ)

英数字, 英字, 数字	isalnum(c), isalpha(c), isdigit(c)
英大文字, 英小文字	isupper(c), islower(c)
16 進数文字(a-f,A-F,0-9)	isxdigit(c)
制御文字(0x1F-0x7F)	iscntrl(c)
表示可能文字(space 含),(space 以外)	isprint(c), isgraph(c)
表示可能文字(英数・space 以外)	ispunct(c)
スペース文字(space,¥n,¥r,¥f,¥t,¥v)	isspace(c)
空白文字(space,¥t)	isblank(c)

文字変換

小文字へ変換, 大文字へ変換	var = tolower(c); toupper(c);

文字列操作 <string.h>

文字列長さ	strlen(str)
コピー(s2→s1),n 文字のみ	strcpy(s1,s2), strncpy(s1,s2,n)
連結 s1+s2→s1	strcat(s1,s2), strncat(s1,s2,n)
比較(0=等/+x=s1>s2)	strcmp(s1,s2), strncmp(s1,s2,n)
文字検索(str から c を先頭から検索)	strchr(str,c)
文字検索(str から c を後方から検索)	strrchr(str,c)
文字列検索(str1 から str2 を検索)	strstr(str1,str2)

日付及び時間 <time.h>

時刻型	clock_t, time_t
時刻構造体	struct tm
メンバ 夏時間フラグ	tm_isdst
1 月 1 日からの日数	tm_yday
日曜からの曜日[0,6]	tm_wday
1900 年からの年数	tm_year
1 月からの月数[0,11]	tm_mon
日[1,31]	tm_mday
時[0,23]	tm_hour
分[0,59]	tm_min
秒[0,60]	tm_sec
clock 関数 1 秒当たりの個数	CLOCKS_PRE_SEC
使用プロセス時間	clock_t clock();
例. clock()/CLOCKS_PER_SEC 使用時間の秒を求める	
現在時刻取得	time_t time(*time_t);
時刻差の秒(time2-time1)	double difftime(time2,time1)
時刻変換 UTC 時刻, 地方時刻	tm gmtime(time_t),localtime()
暦時刻へ変換	mktime(*tm)
時刻文字列変換(Sun Sep 16 01:03:52 1973¥n¥0)	asctime(*tm)
書式付き時刻文字列変換	strftime(str,n,format,*tm)

数学関数 <math.h>

三角関数 sin(double x),cos(),tan(),asin(),acos(),atan() atan2(x,y),sinh(),cosh(),tanh(),asinh(),acosh()atanh()	
指数関数 e^x, 2^x, $y \times 2^{exp}$	exp(x),exp2(x), ldexp(y, exp)
指数部と仮数部の分離	frexp(val, *exp)
対数関数 $\log_e x$,$\log_{10} x$,$\log_2 x$	log(x), log10(x), log2(x)
整数小数の分離(返値小数部,ptr 整数部)	modf(var, *ptr)
浮動小数点剰余	fmod(x,y)
べき関数 x^y,$\sqrt{x}$, $\sqrt[3]{x}$	pow(x,y),sqrt(x),cbrt()
$\sqrt{x^2+y^2}$	hypot(x,y)
絶対値	fabs(x)
最近設整数関数	ceil(x), floor(x), rouond(x)
ガンマ関数誤差関数,余誤差関数	erf(x),erfc(x)
大きい方の値,小さい方の値	fmax(x,y), fmin(x,y)

double 関数は、上記の関数で引数・返値ともに double 型

float 型関数は最後に f がつく関数群 ex. fabsf(float x)

long double 型関数は最後に l がつく関数群 ex. sinl(x)

その他ヘッダ

ワイド文字拡張,操作	<wchar.h>, <wctype.h>
共通定義,可変引数	<stddef.h>, <stdarg.h>
複素数計算	<complex.h>
型総称型数学関数	<tgmath.h>
浮動小数点環境,型特性	<fenv.h>, <float.h>
整数型の大きさ	<limits.h>
診断機能,エラー,シグナル	<assert.h>, <errno.h>, <signal.h>

* 色分け：C89/C90 C95 C99

ご意見・コメントは Twitter: @cegajp / Mail: info@cega.jp へ