Cheatography

C programming Cheat Sheet

by dali_orbaj via cheatography.com/202287/cs/42926/

Bloques

switch	switch(argumento){case x: (...);break;default:break;}	El argumento es una expresion entera y los cases deben ser diferentes. El default es opcional
for	for (init;cond;post)	Es equivalente a: init; while(cond){(...)post;}

Parametros para printf()

%d/i	int
%f	double (usando punto decimal)
%x	hexadecimales (int sin signo en base 10)
%o	octales (int sin signo ni cero inicial en base 10)
%u	int sin signo en base 10
%c	Muestra el caracter que simboliza un int
%s	Cadena de caracteres
%e	double (decimal en notacion exponencial)
%g	double (decimal con la notacion que requiera menor espacio)
%%	imprime %
%p	direccion de memoria (puntero)

Entrada y salida de datos

printf()	Imprime por pantalla
scanf()	Lee datos y los guarda en la variable proporcionada como argumento
getchar()	Lee cadenas de caracteres uno a uno
putchar()	Imprime un caracter por pantalla
gets()	Lee una linea de stdin (hasta \n) y la guarda en el buffer
puts()	Imprime una cadena con un \n al final

static

Dentro de una funcion	No pierde su valor entre invocación e invocación.
Fuera de una función	Visible nomas en el entorno donde está definida, compartida en ese .c.

extern

Dentro de una funcion	Solo esa función va a poder acceder a la variable.
Fuera de una funcion	Visible desde todos los archivos del programa.

(!) Para declarar (no definir) el uso de una variable que es de otro código.

Punteros

Definicion	tipoApuntado * nombrePuntero;
Parametro	tipo nombreFunc (tipo * nombreParam, ...);
Invocacion	nombreFunc(&variable);

Punteros (cont)

	nombreFunc(variablePunt);
Equivalentes	arreglo <--> &arreglo[0]
	*arreglo <--> arreglo[0]

<string.h>

unsigned int strlen(const char * s);	Recorre el vector y devuelve la longitud sin contar el 0 final.
char * strcpy(char * t, const char * s);	Copia todos los caracteres de source a target. Asume que hay suficiente lugar y que source es null terminated. Devuelve un puntero a la cadena target (la direccion donde copio).
char * strncpy(char * t, const char * s, int n);	Copia hasta n caracteres y no pone el cero final si n es menor a la cantidad de caracteres que tiene que copiar. Si N es mayor copia todo y completa con ceros hasta llegar a N caracteres. Retorna una copia de target.
char * strcat-(char * t, const char * s);	Concatena, agrega source al final de target
char * strncat(char * t, const char * s, int n);	Concatenar hasta n caracteres. Agrega como máximo n caracteres de la matriz de caracteres apuntada por s, deteniéndose si se encuentra el carácter nulo, al final de la cadena de bytes null terminated apuntada por target. El carácter s[0] reemplaza al cero al final de target. El carácter nulo de terminación siempre se agrega al final (por lo que el número máximo de bytes que la función puede escribir es n + 1).
int strcmp(const char * t, const char * s);	Compara los valores ASCII de 2 cadenas de bytes null-terminated haciendo t-s
	-Valor negativo si t aparece antes que s. -Cero si t y s son iguales. -Valor positivo si aparece t después que s en la tabla ascii.
int strncm-p(const char * t, const char * s, int n);	Compara los valores ascii de t y s pero hasta N caracteres.
char * strchr(const char * s, char c);	Devuelve la dirección de memoria de la primera aparición del carácter c. Si no aparece devuelve null. El carácter de termin-ación se considera parte de la cadena y se puede encontrar al buscar '\ 0'.
char * strrchr(const char * s, char c);	Lo mismo pero empezando por la derecha.

<string.h> (cont)	
char * strstr(const char * t, const char * s);	Devuelve, si el string s está contenido en t, la dirección del primer carácter de s en t.
char * strpbrk(const char * s, const char * set);	Busca la primera aparición de alguno de los caracteres, el primero que aparezca.
int strcasecmp (const char *, const char *);	Compara 2 caracteres pero a diferencia de strcmp, ignora minusculas y mayusculas, entonces si pongo a y A devuelve que son iguales.
int strncasecmp (const char *, const char *, size_t);	Lo mismo pero hasta N caracteres.

Vectores	
Arreglo	tipo nombreAreglo [dimension] = {valor0, valor1, valor2, …}. De tipo int por default.
	Parametro/prototipo: const/- tipo nombreVector[]. Argumento: nombreVector.
Matriz	tipo nombreMatriz[filas][cols]={{valor 0, valor1, ...}, {valor0, valor1, ...}, ...}.
	Parametro/prototipo: const/- tipo nombreMatriz[][COLS]. Argumento: nombreMatriz.

Structs	
Definicion	struct nombreRegistro{ tipo1 nombreCampo1; tipo2 nombreCampo2; ... tipoN nombreCampoN; };
Declaracion variable de tipo struct	struct nombreRegistro { ... } nombreVariable;
	struct nombreRegistro { ... }; struct nombreRegistro nombreVariable;
Usando typedef	typedef struct nombreOptativo{ ... } nombreTipo; struct nombreOptativo unaVariable; nombreTipo otraVariable; //Estas dos formas de declarar una struct son equivalentes
Inicializacion	struct nombreStruct nombreVariable = {dato1, dato2, ... , datoN};
Acceso al campo de una struct	variableStruct . nombreCampo
Parametro	funcion(nombreVariableDeTipoEstructura);
Retornada en el nombre de una funcion	nombreVariableDeTipoEstructura = funcion();
Asignacion de una estructura a otra	estructura1 = estructura2;
Tamaño de una estructura	Con sizeof
Posicion de un campo	#include <stddef.h> offsetof (tipo , campo)

Structs (cont)

Operador flecha	p->code es equivalente a (*p).code. Sirve cuando modifico una estructura que recibo como parametro en una funcion, que se recibe un puntero a estructura.
Bit fields	Para indicar cuántos bits ocupa cada campo entero. Ej: unsigned int tamaño: 6; // 6 bits

(!) Una struct como parametro de una funcion: Se envía al stack una copia de la estructura. Como se envía una copia, si la función modifica un campo, la variable original (el parámetro actual) no se ve alterada.
(!) Las estructuras no puede ser comparadas

Puntero a funcion

Parametro	elemType (*function) (elemType)
Argumento	function
Arreglo de funciones	elemType (*arrayName[]) (elemType) = {sin, cos, etc};
Ejecucion de funcion en arreglo	arrayName[index] (data);
Ejecucion de funcion	function(data);

(!) "data" lease como el argumento

Constantes simbolicas predefinidas

__LINE__	Constante decimal con el nro. de la linea actual
__FILE__	String que contiene el nombre del archivo que se esta compilando
__DATE__	String con la fecha de compilación
__TIME__	String con la hora de compilación
__func__	String con el nombre de la funcion

<math.h>

double fabs(double x);	valor absoluto de x
double floor(double x);	entero más grande menor o igual a x
double ceil(double x);	valor entero más pequeño mayor o igual a x
double fmod(double x,double y);	resto de x dividido por y
double sqrt(double x);	raíz cuadrada de x
double pow(double x,double y);	x a la y
double exp(double x);	e a la x
double log(double x);	logaritmo natural (en base e) de x
double log10(double x);	logaritmo en base 10 de x
double sin(double radians);	sin
double cos(double radians);	cos
double tan(double radians);	tan

<ctype.h>

<ctype.h> (cont)

int isalnum(int c);	es digito o letra
int isprint(int c);	tiene representacion visual en tabla ascii. 0 si no se puede imprimir
int ispunct(int c);	!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
int isspace(int c);	espacio, tabs, newline, etc
int toupper(int c);	
int tolower(int c);	
int iscntrl (int c)	
int isgraph (int c)	

(!) Devuelve 0 o un valor distinto de 0
(!) Ninguno cambia el valor de la variable argumento

<stdlib.h>

int islower(int c);	
int isupper(int c);	
int isalpha(int c);	es letra
int isdigit(int c);	de 0 a 9
int isxdigit(int c);	es digito hexadecimal

int abs(int num);	Modulo
long labs(long num);	Módulo para long’s
int rand(v-oid);	Valor pseudo aleatorio, le paso time(NULL)
void srand(-unsigned int seed);	Setea el valor de inicio de rand
exit(0)	lo mismo que return 0, nomas p/el main con un error irrecuperable
double atof(const char * s);	Devuelve lo que lee del string s en un double.
int atoi(const char * s);	Devuelve lo que lee del string s en un entero. No redondea. Y si es un numero mas grande de lo que puede representar, va a devolver cualquier cosa por el overflow.
long atol(const char * s);	Devuelve lo que lee del string s en un long.

<stdlib.h> - Memoria dinamica

void * malloc-(size_t size);	Reserva memoria. Ejemplo: int *v = malloc(20*sizeof(i-nt)). La función malloc retorna en su nombre la dirección de una zona de memoria de size bytes reservada en forma dinámica, setea errno en ENOMEM si no hay memoria libre.
void * calloc-(size_t nobj, size_t size);	Además de reservar la memoria inicializa el vector con ceros. La función calloc retorna en su nombre la dirección de una zona de memoria de size x nobj bytes reservada en forma dinámica e inicializada en cero
void *reallo-c(void* p, size_t size);	Función para pedir que agrande o achique el vector. La función realloc modifica el tamaño de una zona de memoria previamente reservada que comienza en la dirección p
void free(void * p);	La función free libera la zona de memoria previamente reservada que comienza en la dirección p

typedef unsigned int size_t;

(!) void * significa que es un puntero de cualquier tipo, es decir un puntero genérico, se le puede asignar el valor de cualquier tipo de puntero y, asimismo, a un puntero de cualquier tipo se le puede asignar un puntero generico. Lo que no se puede hacer con un puntero generico es desreferenciarlo, ya que estaria determinando la cantidad de bytes que ocupa cada tipo de dato, para desreferenciar, antes habria que castear a algun tipo de puntero --> * (int *) p;

<stdbool.h>

bool	true=1 y false=0

<stdio.h>

int getchar(void);

int putchar(int c);

int ungetc(int c, FILE * stream);

int printf(const char *fmt, ...);

int puts(const char *s);

void clearerr(FILE * stream);

int feof(FILE * stream);

int ferror(FILE * stream);

void perror(const char * s);

int sprintf(char * s, const char * fmt, ...);	La única diferencia con printf es que en lugar de enviarlo a la salida estándar lo almacena en un string.
int sscanf(char * s, const char * fmt, ...);	En lugar de leer el texto de la entrada estándar lo toma del string. Y devuelve la cantidad de elementos que se asignaron correctamente.

(!) Todas estas funciones para string se encargan de poner un cero al final menos strncopy

Precedencia y asociatividad

() [] . ->	Izq a der
! ~ ++ -- + _ (tipo) sizeof	Der a izq
* / %	Izq a der
+ -	Izq a der
<< >>	Izq a der
< <= > >=	Izq a der
== !=	Izq a der
&	Izq a der
^	Izq a der
|	Izq a der
&&	Izq a der
||	Izq a der
?:	Der a izq
= += -= *= /= %= &= ^= |= <<= >>=	Der a izq

Cheatography

C programming Cheat Sheet

by dali_orbaj via cheatography.com/202287/cs/42926/

Precedencia y asociatividad (cont)

,	Izq a der

(!) Los +, -, * unarios tienen mayor precedencia que las formas binarias

Operadores p/bits

~	Complemento
^	Xor
&	And
|	Or
Sizeof	#bits que ocupa un tipo de dato

Constantes

Long	123L/l
Unsigned	123U/u
Unsigned long	123UL/ul
Float	12.3F/f
Double	12.3
Long double	12.3L/l
Octales	Prefijo 0
Hexadecimales	Prefijo 0x

Datos

char	8	-128 a 127
unsigned char	8	0 a 255
signed char	8	-128 a 127
short	16	-32768 a 32767
int	16	-32768 a 32767
unsigned int	16	0 a 65535
signed int	16	-32768 a 32767
short int	16	-32768 a 32767
unsigned short int	16	0 a 65535
signed short int	16	-32768 a 32767
long int	32	-2147483648 a 2147483647
signed long int	32	-2147483648 a 2147483647
unsigned long int	32	0 a 4294967295
long	32	-2147483648 a 2147483647
unsigned long	32	0 a 4294967295

Datos (cont)

float	32	3.4E-38 a 3.4E+38
double	64	1.7E-308 a 1.7E+308
long double	64/80	1.7E-308 a 1.7E+308 o 3.4E-4932 a 1.1E+4932

(!) char sin default e int signed y short por default.

Redireccionamiento

<	Entrada estandar
<2	Entrada por error
>	Salida estandar
2>	Salida por error

Especificaciones de conversion para scanf

%*d	Se lee un entero pero sin guardarlo en ninguna variable.
%3u	Longitud maxima, unsigned 3
%ho	la h quiere decir que es un short y la o que es octal
%*x	
%4f	
%*c	
%5s	

Listas

Definicion	typedef struct node * TList; typedef struct node { int elem; struct node * tail; } TNode;
Argumento a funcion	double (*func) (double)

(!) Una lista vacía se representa con el valor NULL