Programación de Sistemas CCPG1008

Federico Domínguez, PhD.

Unidad 1 – Sesión 4: Introducción a C

Introducción a C

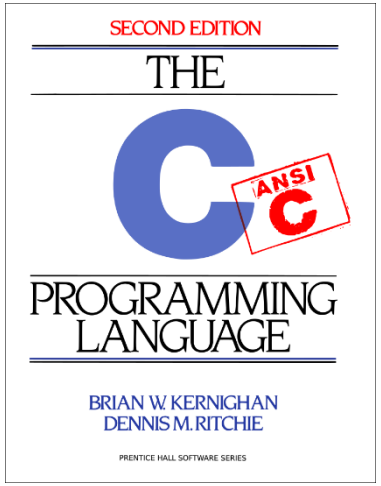
C fue desarrollado por Dennis Ritchie en Bell Labs entre 1969 y 1973 con el propósito de reescribir UNIX.

C es un lenguaje compilado de bajo nivel.

C es un lenguaje imperativo (operaciones secuenciales que cambian el estado del sistema) y fuertemente tipado (el tipo de datos de una variable es definido en el lenguaje).

C ha sido estandarizado con el tiempo:

- ANSI C: Estándar por la *American National Standards Institute* (ANSI)
- C89, primer estándar definido por la ANSI en 1989
- C99, estándar ISO definido en 1999 y adoptado por la ANSI
- C11, el último estándar definido en 2011. ISO y ANSI



Introducción a C

C se basa en el uso de funciones (parte del paradigma de programación estructurada).

Utiliza la función *main* como entrada.

```
/* Hello World program */

#include<stdio.h>

int main()
{
    printf("Hello World\n");
}
```

Introducción a C

```
#include <stdio.h>                    include information about standard library
main()                                          define a function called main
                                              that received no argument values
{                                      statements of main are enclosed in braces
    printf("hello, world\n");                main calls library function printf
                                          to print this sequence of characters
}                                          \n represents the newline character
```

The first C program

```c
#include <stdio.h>

/* print Fahrenheit-Celsius table
for fahr = 0, 20, ..., 300 */

int main()
{
        int fahr, celsius;
        int lower, upper, step;
        lower = 0; /* lower limit of temperature scale */
        upper = 300; /* upper limit */
        step = 20; /* step size */
        fahr = lower;

        while (fahr <= upper) {
                celsius = 5 * (fahr-32) / 9;
                printf("%d\t%d\n", fahr, celsius);
                fahr = fahr + step;
        }
}
```

Ingreso de datos en C

```c
#include <stdio.h>

int main()
{
        char str[10];
        printf("Ingrese su nombre: ");
        scanf("%s", str);

        printf("Su nombre es: %s\n", str);
}
```

Directiva para el preprocesador del compilador: Reemplaza en el código la palabra SIZE con el número 20.

```c
#include <stdio.h>

#define SIZE 20

void hacerAlgo(char *);

int main()
{
        char str[SIZE];
        printf("Ingrese su nombre: ");
        scanf("%s", str);

        hacerAlgo(str);
}

void hacerAlgo(char *str)
{
    for(int i=0; i<SIZE; i++)
    {
        char c = str[i];

        if(c == 0)
                break;

        printf("%c ",c);
    }
}
```

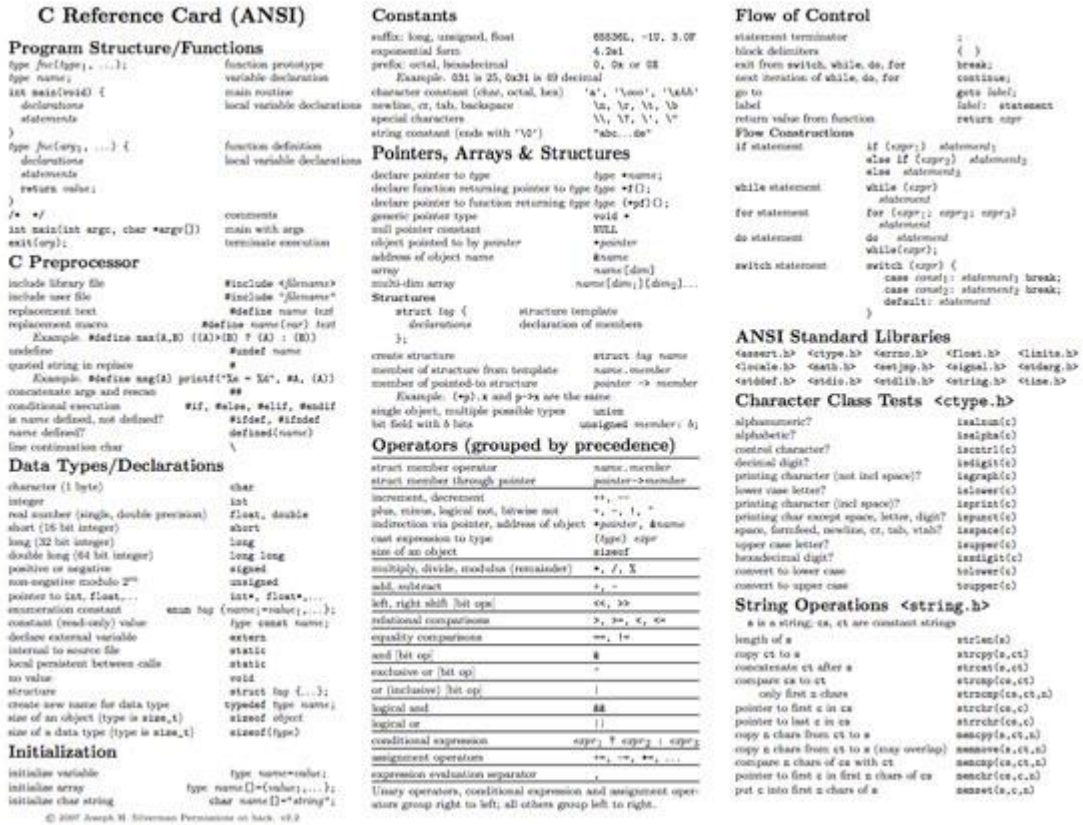
*Antes de poder usar una función, es necesario declarar su **prototipo**.*

*La función `hacerAlgo` es **llamada** aquí.*

*La función `hacerAlgo` es **implementada** aquí.*

Ejercicio: Modificar código anterior para contar número de dígitos y letras en el texto ingresado.

Usar C *cheat sheet* provista …



C Reference Card (ANSI)

Program Structure/Functions

type func(type1, ...);	function prototype
type name;	variable declaration
int main(void) {	main routine
declarations	local variable declarations
statements	
}	
type func(arg1, ...) {	function definition
declarations	local variable declarations
statements	
return value;	
}	
/* */	comments
int main(int argc, char *argv[])	main with args
exit(arg);	terminate execution

C Preprocessor

include library file	#include <filename>
include user file	#include "filename"
replacement text	#define name text
replacement macro	#define name(var) text
Example. #define max(A,B) ((A)>(B) ? (A) : (B))	
undefine	#undef name
quoted string in replace	#
Example. #define dag(A) printf("%s = %d", #A, (A))	
concatenate args and rescan	##
conditional execution	#if, #else, #elif, #endif
is name defined, not defined?	#ifdef, #ifndef
name defined?	defined(name)
line continuation char	\

Data Types/Declarations

character (1 byte)	char
integer	int
real number (single, double precision)	float, double
short (16 bit integer)	short
long (32 bit integer)	long
double long (64 bit integer)	long long
positive or negative	signed
non-negative modulo 2^m	unsigned
pointer to int, float,...	int*, float*,...
enumeration constant	enum tag {name1=value1,...};
constant (read-only) value	type const name;
declare external variable	extern
internal to source file	static
local persistent between calls	static
no value	void
structure	struct tag {...};
create new name for data type	typedef type name;
size of an object (type is size_t)	sizeof object
size of a data type (type is size_t)	sizeof(type)

Initialization

initialize variable	type name=value;
initialize array	type name[]={value1,...};
initialize char string	char name[]="string";

Constants

suffix: long, unsigned, float	65536L, 10U, 3.0F
exponential form	6.2e1
prefix: octal, hexadecimal	0, 0x or 0X
Example. 031 is 25, 0x31 is 49 decimal	
character constant (char, octal, hex)	'a', '\ooo', '\xhh'
newline, cr, tab, backspace	\n, \r, \t, \b
special characters	\\, \?, \', \"
string constant (ends with '\0')	"abc...de"

Pointers, Arrays & Structures

declare pointer to type	type *name;
declare function returning pointer to type	type *f();
declare pointer to function returning type	type (*pf)();
generic pointer type	void *
null pointer constant	NULL
object pointed to by pointer	*pointer
address of object name	&name
array	name[dim]
multi-dim array	name[dim1][dim2]...

Structures

struct tag {	structure template
declarations	declaration of members
};	
create structure	struct tag name
member of structure from template	name.member
member of pointed-to structure	pointer -> member
Example. (*p).x and p->x are the same	
single object, multiple possible types	union
bit field with b bits	unsigned member: b;

Operators (grouped by precedence)

struct member operator	name.member
struct member through pointer	pointer->member
increment, decrement	++, --
plus, minus, logical not, bitwise not	+, -, !, ~
indirection via pointer, address of object	*pointer, &name
cast expression to type	(type) expr
size of an object	sizeof
multiply, divide, modulus (remainder)	*, /, %
add, subtract	+, -
left, right shift [bit op]	<<, >>
relational comparisons	>, >=, <, <=
equality comparisons	==, !=
and [bit op]	&
exclusive or [bit op]	^
or (inclusive) [bit op]	|
logical and	&&
logical or	||
conditional expression	expr1 ? expr2 : expr3
assignment operators	+=, -=, *=, ...
expression evaluation separator	,

Unary operators, conditional expression and assignment operators group right to left; all others group left to right.

Flow of Control

statement terminator	;
block delimiters	{ }
exit from switch, while, do, for	break;
next iteration of while, do, for	continue;
go to	goto label;
label	label: statement
return value from function	return expr

Flow Constructions

if statement	if (expr1) statement1 else if (expr2) statement2 else statement3
while statement	while (expr) statement
for statement	for (expr1; expr2; expr3) statement
do statement	do statement while(expr);

switch statement	switch (expr) {
	case const1: statement1; break;
	case const2: statement2; break;
	default: statement
	}

ANSI Standard Libraries

<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>

Character Class Tests <ctype.h>

alphanumeric?	isalnum(c)
alphabetic?	isalpha(c)
control character?	iscntrl(c)
decimal digit?	isdigit(c)
printing character (not incl space)?	isgraph(c)
lower case letter?	islower(c)
printing character (incl space)?	isprint(c)
printing char except space, letter, digit?	ispunct(c)
space, formfeed, newline, cr, tab, vtab?	isspace(c)
upper case letter?	isupper(c)
hexadecimal digit?	isxdigit(c)
convert to lower case	tolower(c)
convert to upper case	toupper(c)

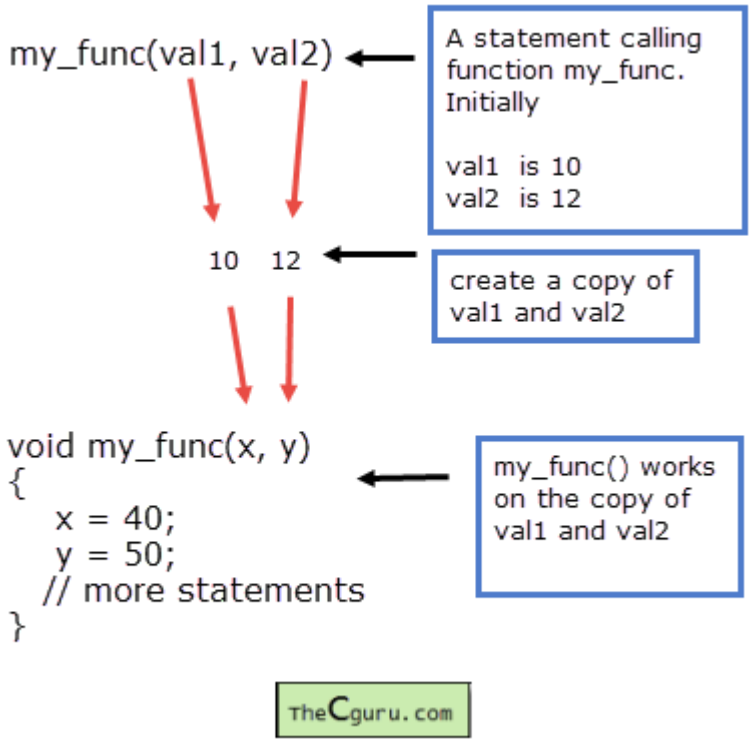
String Operations <string.h>

s is a string; cs, ct are constant strings

length of s	strlen(s)
copy ct to s	strcpy(s,ct)
concatenate ct after s	strcat(s,ct)
compare cs to ct	strcmp(cs,ct)
only first n chars	strncmp(cs,ct,n)
pointer to first c in cs	strchr(cs,c)
pointer to last c in cs	strrchr(cs,c)
copy n chars from ct to s	memcpy(s,ct,n)
copy n chars from ct to s (may overlap)	memmove(s,ct,n)
compare n chars of cs with ct	memcmp(cs,ct,n)
pointer to first c in first n chars of cs	memchr(cs,c,n)
put c into first n chars of s	memset(s,c,n)

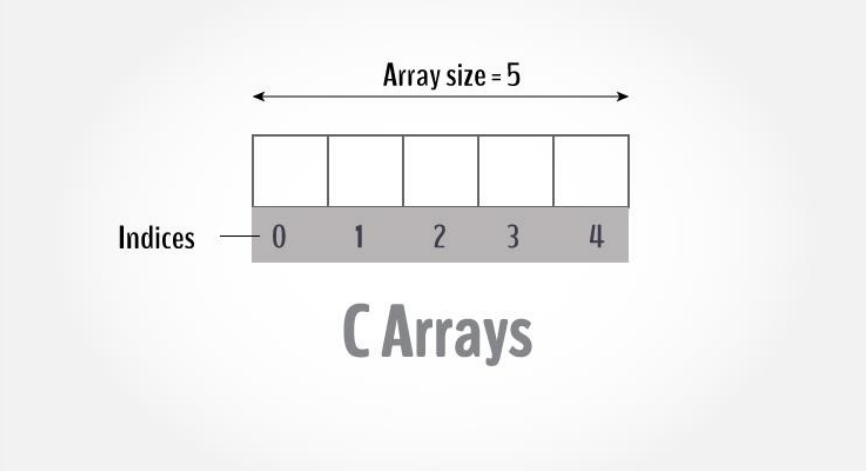
Otras características de C

Parámetros son pasados por valor…

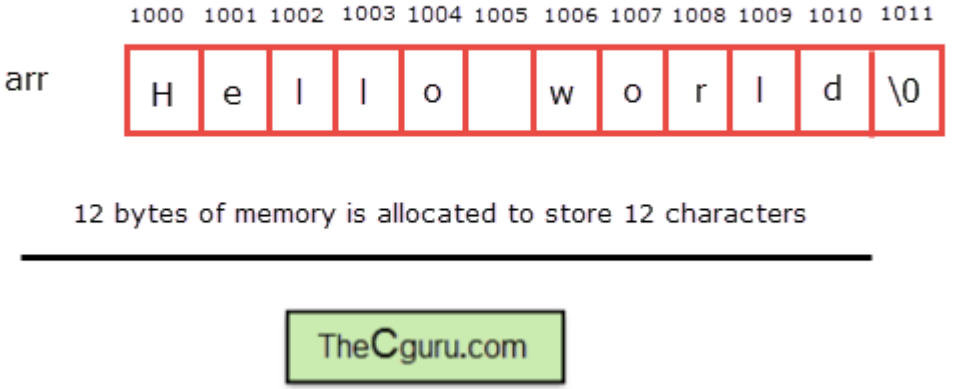


Otras características de C

Indexación de arreglos siempre empieza desde cero…



Arreglo de caracteres siempre terminan con el carácter *null* (\0)



Para la próxima clase

Lectura:

- Capítulos 1 y 2 de **The C programming Language** (Brian W. Kernighan and Dennis M. Ritchie) 2da edición

Tarea

- Crear una cuenta en Bitbucket (bibucket.org)

Práctica:

- Programación en C