Caracteres y Cadenas

Fundamentos de Programación

Conceptos Básicos

- Caracteres
  - Valor entero representado como caracter entre comillas simples. Por ejemplo: 'z' representa al valor entero de z
  - Internamente se representa como un tipo de dato enumerado usando el código ASCII (*código estándar americano para el intercambio de información*).
- Cadenas
  - Es un arreglo de caracteres que:
    - Puede incluir letras, dígitos y caracteres especiales (*, /, $)
    - Tiene un puntero al primer caracter
    - Cuyo valor de la cadena es la dirección de memoria del primer elemento.

Propiedades Importantes del Código ASCII

1. Los códigos para los caracteres que representan dígitos del 0 al 9 son consecutivos.
2. Las letras en el alfabeto están divididos en dos rangos: uno para las mayúsculas (A-Z) y otro para las minúsculas (a-z). Sin embargo dentro de cada rango los valores ASCII son consecutivos.

Constantes de Tipo Caracter

- Es un estándar para referirse a un carácter específico en C.
- Para referirse al código ASCII de la letra A, se especifica ‘**A**’, el cual es el 65.
- Para referirse al código del carácter 9, de forma similar, **‘9’.**

CUIDADO: El referirse al carácter, no es lo mismo que referirse al valor entero. El 9 es diferente del ‘9’.

Operaciones con Caracteres

Se puede:

- Sumar un entero a un carácter
- Restar un entero de un caracter
- Restar un caracter de otro
- Comparar dos caracteres entre sí

CUIDADO: Al sumar o restar el resultado no debe salirse del rango de representación ASCII

Manejo de Cadenas

- **Definición**
  - Como un arreglo de caracteres o una variable de tipo `char *`

```
char color[] = "blue";
char *colorPtr = "blue";
```

  - Una cadena se representa como un arreglo de caracteres y termina con `'\0'`
    - `color ("blue")` tiene 5 elementos
- **Lectura**
  - Utilizando `scanf`

```
scanf("%s", cadena);
```

    - Copia la entrada en el arreglo `cadena[]`
    - No se necesita el **&** (porque una cadena es un puntero)
  - Recuerde dejar espacio en el arreglo para el fin de cadena`'\0'`
- **Mostrar por pantalla**
  - Utilizando `printf`

```
printf("%s",cadena);
```

Ejemplos

```
char RandomLetra(void)
{
   return (RandomInteger (‘A’, ‘Z’));
}
```

```
Int esMayuscula (char ch)
{
   if (ch >= ‘A’ && ch <=‘Z’)
      return 1;
  return 0;
}
```

```
Int esDigito (char ch)
{
   if(ch >= ‘0’ && ch <=‘9’)
      return 1;
    return 0;
}
```

```
Int esMinuscula (char ch)
{
  if(ch >= ‘a’  &&  ch <=‘z’)
    return 1;
  return 0;
}
```

Interfaces útiles

La interfaz **ctype.h**

- Contiene un gran número de funciones para determinar el tipo de carácter, entre las principales tenemos:

- *islower(ch)* retorna TRUE si el carácter ch es minúscula
- *isupper(ch)* retorna TRUE si el carácter ch es mayúscula
- *isalpha(ch)* retorna TRUE si ch es un valor alfabético
- *isdigit(ch)* retorna TRUE si ch es un dígito
- *isalnum(ch)* retorna TRUE si ch es un valor alfanumérico
- *ispunct(ch)* retorna TRUE si ch es un símbolo de puntuación
- *isspace(ch)* retorna TRUE si ch es un carácter en blanco

ctype.h: Librería de manejo de caracteres

Prototype	Description
`int isdigit( int c );`	Returns `true` if `c` is a digit and `false` otherwise.
`int isalpha( int c );`	Returns `true` if `c` is a letter and `false` otherwise.
`int isalnum( int c );`	Returns `true` if `c` is a digit or a letter and `false` otherwise.
`int isxdigit( int c );`	Returns `true` if `c` is a hexadecimal digit character and `false` otherwise.
`int islower( int c );`	Returns `true` if `c` is a lowercase letter and `false` otherwise.
`int isupper( int c );`	Returns `true` if `c` is an uppercase letter; `false` otherwise.
`int tolower( int c );`	If `c` is an uppercase letter, `tolower` returns `c` as a lowercase letter. Otherwise, `tolower` returns the argument unchanged.
`int toupper( int c );`	If `c` is a lowercase letter, `toupper` returns `c` as an uppercase letter. Otherwise, `toupper` returns the argument unchanged.
`int isspace( int c );`	Returns `true` if `c` is a white-space character—newline (`'\n'`), space (`' '`), form feed (`'\f'`), carriage return (`'\r'`), horizontal tab (`'\t'`), or vertical tab (`'\v'`)—and `false` otherwise
`int iscntrl( int c );`	Returns `true` if `c` is a control character and `false` otherwise.
`int ispunct( int c );`	Returns `true` if `c` is a printing character other than a space, a digit, or a letter and `false` otherwise.
`int isprint( int c );`	Returns `true` value if `c` is a printing character including space (`' '`) and `false` otherwise.
`int isgraph( int c );`	Returns `true` if `c` is a printing character other than space (`' '`) and `false` otherwise.

Stdlib.h: Librería de funciones de conversión

- Convierte cadenas de dígitos a enteros y valores de punto flotante.

Function prototype	Function description
`double atof( const char *nPtr );`	Converts the string `nPtr` to `float`.
`int atoi( const char *nPtr );`	Converts the string `nPtr` to `int`.
`long atol( const char *nPtr );`	Converts the string `nPtr` to `long int`.
`double strtod( const char *nPtr, char **endPtr );`	Converts the string `nPtr` to `double`.
`long strtol( const char *nPtr, char **endPtr, int base );`	Converts the string `nPtr` to `long`.
`unsigned long strtoul( const char *nPtr, char **endPtr, int base );`	Converts the string `nPtr` to `unsigned long`.

stdio.h

Function prototype	Function description
`int getchar( void );`	Inputs the next character from the standard input and returns it as an integer.
`char *gets( char *s );`	Inputs characters from the standard input into the array `s` until a newline or end-of-file character is encountered. A terminating null character is appended to the array.
`int putchar( int c );`	Prints the character stored in `c`.
`int puts( const char *s );`	Prints the string `s` followed by a newline character.
`int sprintf( char *s, const char *format, ... );`	Equivalent to `printf`, except the output is stored in the array `s` instead of printing it on the screen.
`int sscanf( char *s, const char *format, ... );`	Equivalent to `scanf`, except the input is read from the array `s` instead of reading it from the keyboard.

String.h: Librería de manipulación de cadenas

- Incluye funciones para:
  - Manipular cadenas
  - Búsqueda en cadenas
  - Manejo de tokens
  - Determine la longitud de cadenas

Function prototype	Function description
`char *strcpy( char *s1, const char *s2 )`	Copies string `s2` into array `s1`. The value of `s1` is returned.
`char *strncpy( char *s1, const char *s2, size_t n )`	Copies at most `n` characters of string `s2` into array `s1`. The value of `s1` is returned.
`char *strcat( char *s1, const char *s2 )`	Appends string `s2` to array `s1`. The first character of `s2` overwrites the terminating null character of `s1`. The value of `s1` is returned.
`char *strncat( char *s1, const char *s2, size_t n )`	Appends at most `n` characters of string `s2` to array `s1`. The first character of `s2` overwrites the terminating null character of `s1`. The value of `s1` is returned.

Funciones de comparación de cadenas

`int strcmp( const char *s1, const char *s2 );`

- Compara string `s1` con `s2`
- Retorna:
  - Un número negativo si `s1 < s2`
  - `Cero`,si `s1 == s2`
  - `Un número positivo` si `s1 > s2`

`int strncmp(const char *s1,const char *s2,size_t n);`

- Compara n caracteres de `s1` en `s2`
- Retorna valores como los anteriores

Funciones de Búsqueda

Function prototype	Function description
`char *strchr( const char *s, int c );`	Locates the first occurrence of character `c` in string `s`. If `c` is found, a pointer to `c` in `s` is returned. Otherwise, a `NULL` pointer is returned.
`size_t strcspn( const char *s1, const char *s2 );`	Determines and returns the length of the initial segment of string `s1` consisting of characters not contained in string `s2`.
`size_t strspn( const char *s1, const char *s2 );`	Determines and returns the length of the initial segment of string `s1` consisting only of characters contained in string `s2`.
`char *strpbrk( const char *s1, const char *s2 );`	Locates the first occurrence in string `s1` of any character in string `s2`. If a character from string `s2` is found, a pointer to the character in string `s1` is returned. Otherwise, a `NULL` pointer is returned.
`char *strrchr( const char *s, int c );`	Locates the last occurrence of `c` in string `s`. If `c` is found, a pointer to `c` in string `s` is returned. Otherwise, a `NULL` pointer is returned.
`char *strstr( const char *s1, const char *s2 );`	Locates the first occurrence in string `s1` of string `s2`. If the string is found, a pointer to the string in `s1` is returned. Otherwise, a `NULL` pointer is returned.
`char *strtok( char *s1, const char *s2 );`	A sequence of calls to `strtok` breaks string `s1` into "tokens"—logical pieces such as words in a line of text—separated by characters contained in string `s2`. The first call contains `s1` as the first argument, and subsequent calls to continue tokenizing the same string contain `NULL` as the first argument. A pointer to the current token is returned by each call. If there are no more tokens when the function is called, `NULL` is returned.