Universidad Nacional Autónoma de México

Facultad de Ingeniería

División de Ciencias Básicas

Estructura de Datos y Algoritmos I

Alumno: _Bear Almaraz Miguel Ángel

Semestre 2021-2

Nombre de la actividad:

Actividad 2: Acordeón C

Fecha: 08/06/2021

Acordeón C

Para crear un programa en C se siguen tres etapas principales: edición, compilación y ejecución.

Edición: Se escribe el código fuente en lenguaje C desde algún editor de textos.

Compilación: A partir del código fuente (lenguaje C) se genera el archivo en lenguaje máquina (se crea el programa objeto o ejecutable).

Ejecución: El archivo en lenguaje máquina se puede ejecutar en la arquitectura correspondiente

Comentarios

El comentario por línea inicia cuando se insertan los símbolos '//' y termina con el salto de línea (hasta donde termine el renglón)

El comentario por bloque inicia cuando se insertan los símbolos '/*' y termina cuando se encuentran los símbolos '*/'. Cabe resaltar que el comentario por bloque puede abarcar varios renglones.

Declaración de variables

Para declarar variables en C se sigue la siguiente sintaxis:

[modificadores] tipoDeDato identificador [= valor];

Tipos de datos

Los tipos de datos básicos en C son:

Caracteres: codificación definida por la máquina.

Enteros: números sin punto decimal.

Flotantes: números reales de precisión normal.

Dobles: números reales de doble precisión.

Las variables enteras que existen en lenguaje C son:

Tipo	*Bits*	*Valor Mínimo*	*Valor Máximo*
signed char	8	-128	127
unsigned char	8	0	255
signed short	16	-32 768	32 767
unsigned short	16	0	65 535
signed int	32	-2,147,483,648	2 147 483 647
unsigned int	32	0	4 294 967 295
signed long	64	9 223 372 036 854 775 808	9 223 372 036 854 775 807
unsigned long	64	0	18 446 744 073 709 551 615
enum	16	-32 768	32 767

Las variables reales que existen en lenguaje C son:

Tipo	*Bits*	*Valor Mínimo*	*Valor Máximo*
float	32	3.4 E-38	3.4 E38
double	64	1.7 E-308	1.7 E308
long double	80	3.4 E-4932	3.4 E4932

Para poder acceder al valor de una variable se requiere especificar el tipo de dato. Los especificadores que tiene lenguaje C para los diferentes tipos de datos son:

Tipo de dato	*Especificador de formato*
Entero	%d, %i, %ld, %li, %o, %x
Flotante	%f, %lf, %e, %g
Carácter	%c, %d, %i, %o, %x
Cadena de caracteres	%s

El especificador de dato se usa para guardar o imprimir el valor de una variable

Operadores

Los operadores aritméticos:

Operador	*Operación*	*Uso*	*Resultado*
+	Suma	125.78 + 62.5	188.28
-	Resta	65.3 - 32.33	32.97
*	Multiplicación	8.27 * 7	57.75
/	División	15 / 4	3.75
%	Módulo	4 % 2	0

Los operadores lógicos:

Operador	*Operación*	*Uso*	*Resultado*
>>	Corrimiento a la derecha	8 >> 2	2
<<	Corrimiento a la izquierda	8 << 1	16
&	Operador AND	5 & 4	4
|	Operador OR	3 | 2	3
~	Complemento ar-1	~2	1

Expresiones lógicas

Operador	*Operación*	*Uso*	*Resultado*
==	Igual que	'h' == 'H'	Falso
!=	Diferente a	'a' != 'b'	Verdadero
<	Menor que	7 < 15	Verdadero
>	Mayor que	11 > 22	Falso
< =	Menor o igual	15 <= 22	Verdadero
> =	Mayor o igual	20 >= 35	Falso

Estructura de control selectiva if

La estructura de control de flujo más simple es la estructura condicional if, su sintaxis es la siguiente:

```
if (expresión_lógica) {
    // bloque de código a ejecutar
}
```

Estructura de control selectiva if-else

La sintaxis de la estructura de control de flujo if-else es la siguiente:

```
if (expresión_lógica) {
    // bloque de código a ejecutar
    // si la condición es verdadera
} else {
    // bloque de código a ejecutar
    // si la condición es falsa
}
```

Estructura de control selectiva switch-case

La sintaxis de la estructura switch-case es la siguiente:

```
switch (opcion_a_evaluar){
    case valor1:
        /* Código a ejecutar*/
    break;
    case valor2:
        /* Código a ejecutar*/
    break;
    …
    case valorN:
        /* Código a ejecutar*/
    break;
    default:
        /* Código a ejecutar*/
}
```

Estructura de control repetitiva while

La estructura repetitiva (o iterativa) while primero valida la expresión lógica y si ésta se cumple (es verdadera) procede a ejecutar el bloque de instrucciones de la estructura, el cual está delimitado por las llaves {}.

```
while (expresión_lógica) {
    // Bloque de código a repetir
    // mientras que la expresión
    // lógica sea verdadera.
}
```

Estructura de control repetitiva do-while

do-while es una estructura cíclica que ejecuta el bloque de código que se encuentra dentro de las llaves y después valida la condición, es decir, el bloque de código se ejecuta de una a ene veces.

```
do {
    /*
    Bloque de código que se ejecuta
    por lo menos una vez y se repite
    mientras la expresión lógica sea
    verdadera.
    */
} while (expresión_lógica);
```

Estructura de control de repetición for

Lenguaje C posee la estructura de repetición for la cual permite realizar repeticiones cuando se conoce el número de elementos que se quiere recorrer. La sintaxis que generalmente se usa es la siguiente:

```
for (inicialización ; expresión_lógica ; operaciones por iteración) {
    /*
        Bloque de código
        a ejecutar
    */
}
```

Break

Algunas veces es conveniente tener la posibilidad de abandonar un ciclo. La proposición break proporciona una salida anticipada dentro de una estructura de repetición, tal como lo hace en un switch. Un break provoca que el ciclo que lo encierra termine inmediatamente

```
#include <stdio.h>

/*
 * Este programa hace una suma de números. Si la suma rebasa la cantidad
 * de 50 el programa se detiene.
 * */

#define VALOR_MAX 5

int main (){
        int enteroSuma = 0;
        int enteroNumero = 0;
        int enteroContador = 0;
        while (enteroContador < VALOR_MAX){
                printf("Ingrese un número:");
                scanf("%d", &enteroNumero);
                enteroSuma += enteroNumero;
                enteroContador++;
                if (enteroSuma > 50){
                        printf("Se rebasó la cantidad límite.\n");
                        break;
                }
```

Continue
La proposición continue provoca que inicie la siguiente iteración del ciclo de repetición que la contiene.

```c
#include <stdio.h>

/*
 * Este programa obtiene la suma de un LIMITE de números pares ingresados
 * */



#define LIMITE 5



int main (){

        int enteroContador = 1;

        int enteroNumero = 0;
        int enteroSuma = 0;
        while (enteroContador <= LIMITE){
                printf("Ingrese número par %d:", enteroContador);
                scanf("%d",&enteroNumero);

                if (enteroNumero%2 != 0){
                        printf("El número insertado no es par.\n");
                        continue;
                }

                enteroSuma += enteroNumero;
                enteroContador++;
        }
```

Arreglos

La sintaxis para definir un arreglo en lenguaje C es la siguiente:

tipoDeDato nombre[tamaño]

Apuntadores

Un apuntador es una variable que contiene la dirección de una variable, es decir, hace referencia a la localidad de memoria de otra variable. Debido a que los apuntadores trabajan directamente con la memoria, a través de ellos se accede con rapidez a un dato.

La sintaxis para declarar un apuntador y para asignarle la dirección de memoria de otra variable es, respectivamente:

TipoDeDato *apuntador, variable;
apuntador = &variable;

Arreglos multidimensionales

Lenguaje C permite crear arreglos de varias dimensiones con la siguiente sintaxis:

tipoDato nombre[tamaño][tamaño]…[tamaño];

Funciones

La sintaxis básica para definir una función es la siguiente:

```
valorRetorno nombre (parámetros){
// bloque de código de la función
}
```

Apuntador a archivo

Los apuntadores a un archivo se manejan en lenguaje C como variables apuntador de tipo FILE que se define en la cabecera stdio.h. La sintaxis para obtener una variable apuntador de archivo es la siguiente:

```
FILE *F;
```

Abrir archivo

La función fopen() abre una secuencia para que pueda ser utilizada y la asocia a un archivo. Su estructura es la siguiente:

```
*FILE fopen(char *nombre_archivo, char *modo);
```

Donde nombre_archivo es un puntero a una cadena de caracteres que representan un nombre válido del archivo y puede incluir una especificación del directorio. La cadena a la que apunta modo determina cómo se abre el archivo.

Cerrar archivo

La función fclose() cierra una secuencia que fue abierta mediante una llamada a fopen().
La firma de esta función es:

```
int fclose(FILE *apArch);
```

Funciones fgets y fputs

Las funciones fgets() y fputs() pueden leer y escribir, respectivamente, cadenas sobre los archivos. Las firmas de estas funciones son, respectivamente:

```
char *fgets(char *buffer, int tamaño, FILE *apArch);
char *fputs(char *buffer, FILE *apArch);
```

Funciones fscanf y fprintf

Las funciones fprintf() y fscanf() se comportan exactamente como printf() (imprimir) y scanf() (leer), excepto que operan sobre archivo. Sus estructuras son:

```
int fprintf(FILE *apArch, char *formato, ...);

int fscanf(FILE *apArch, char *formato, ...);
```

Funciones fread y fwrite

fread y fwrite son funciones que permiten trabajar con elementos de longitud conocida. fread permite leer uno o varios elementos de la misma longitud a partir de una dirección de memoria determinada (apuntador).

El valor de retorno es el número de elementos (bytes) leídos. Su sintaxis es la siguiente:

int fread(void *ap, size_t tam, size_t nelem, FILE *archivo)

Referencias

-http://lcp02.fi-b.unam.mx/