Universidad Nacional Autónoma de México

Facultad de Ingeniería

Asignatura:

Estructura de Datos y Algoritmos I

Actividad #1 | Acordeones

Nombre del Alumno:

Sánchez Estrada Angel Isaac

Fecha:

26/02/2021

ACORDEÓN DEL Lenguaje C

Editores

Un programa en Lenguaje C debe ser escrito en un editor de texto el cual permite copiar, editar, pegar, guardar, etcétera un código; Para después generar un programa ejecutable en la computadora por medio de un compilador.

Compiladores

Ya que se terminó de codificar el programa, es leído y traducido por un programa el cual sirve para transformar un programa entero de un lenguaje de programación a otro. A este programa se le conoce como compilador.

- GCC (GNU Compiler Collection)
    1. Compilación para salida de nombre por defecto a.out (en Windows a.exe)
        - gcc nombre_programa.c
    2. Compilación para salida con algún nombre particular
        - gcc nombre_programa.c -o nombre_programa.out
        - gcc nombre_programa.c -o nombre_programa.exe
    3. Compilación si se incluye una librería especifica
        - gcc nombre_programa c -o nombre_programa -lnombre_libreria

Ejecución

Para la ejecución se tiene considerado que el programa compilado.

1. Para una Ejecución en sistema Unix
    - ./nombre_programa
2. Para una Ejecución en Windows
    - nombre_programa.exe
3. Para una Ejecución con entrada de información por medio de argumentos
    - nombre_programa argumento1 argumento2

Función main

Todo programa escrito en Lenguaje C debe contener la función main dentro de este

```
Int main(){
        return(0);
}
```

Comentarios

// - Comentario de una línea

/**/ - Comentario en bloque o de varias líneas

Declaración de variables
La sintaxis para declarar variables en C es:

- Un identificador
  [modificadores] tipoDeDato identificador [= valor];
- Varios identificadores
  tipoDeDato identificador1[= valor], identificador2[= valor];

Tipos de Datos

- Caracteres: codificación definidas por la maquina
- Enteros: números din punto decimal
- Flotantes: números reales de precisión normal
- Dobles: números reales de doble precisión

Variables Enteras

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

Variables Reales

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

Especificadores de formato

- Entero: %d, %i, %ld, %li, %o, %x
- Flotan: %f, %lf, %e, %g
- Carácter: %c, %d, %i, %o, %x
- Cadena de Caracteres: %s

Identificadores (Reglas)
Es el Nombre con que se almacena en la memoria un dato.

- Debe iniciar con una letra [a-z].
- Puede contener letras [A-Z, a-z], números [0-9] y el carácter guión bajo (_).

Entrada y Salida de Datos

- printf: Muestra en pantalla los datos

```
printf("El valor de la variable real es: %lf", varReal);
```

- scanf: Guarda un dato que el usuario digitó

```
scanf ("%i", &varEntera);
```

- gets: Se usa para cadenas de strings, dado que el scanf solo lee hasta que haya un espacio

- puts: Muestra datos en pantalla, pero solo si está dentro de un condicional

Secuencias de Caracteres de escape

- \a carácter de alarma
- \b retroceso
- \f avance de hoja
- \n salto de línea
- \r regreso de carro
- \t tabulador horizontal
- \v tabulador vertical
- '\0' carácter nulo

Modificadores de alcance

- const: impide que una variable cambie su valor durante la ejecución del programa.
- static: indica que la variable permanece en memoria desde su creación y durante toda la ejecución del programa

Operadores Aritméticos

- + - Suma
- - - Resta
- * - Multiplicación
- / - División
- % - Módulo

Operadores lógicos

- >> - Corrimiento a la derecha
- << - Corrimiento a la izquierda
- & - Operador AND
- | - Operador OR
- ~ - Complemento ar-1

Expresiones lógicas

- == - Igual que
- != - Diferente a
- < - Menor que
- > - Mayor que
- <= - Menor o igual
- >= - Mayor o igual

Condiciones Complejas

- ! - No
- && - Y
- | | - O

Operadores para incrementos y decrementos

- ++ - Agrega una unidad
- -- - Resta una unidad

Estructura de Selección

if

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

If-else

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

Enumeración

```
enum identificador {VALOR1, VALOR2, ... , VALORN};
```

Condicional

```
Condición ? SiSeCumple : SiNoSeCumple
```

switch-case

```
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*/
}
```

Estructuras de Repetición

while

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

do-while

```
do {
        // Bloque de código que se ejecuta
        // por lo menos una vez y se repite
        // mientras la expresión lógica sea verdadera.
```

} while (expresión_lógica);

for

for (inicialización ; expresión_lógica ; operaciones por iteración) {

/*

Bloque de código

a ejecutar

*/

}

Define

#define <nombre> <valor>

Break

Un break provoca que el ciclo que lo encierra termine inmediatamente

Continue

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

Depuración de Programas

- Para depurar un programa que aún no está compilado
  gcc -g -o calculadora calculadora.c

- Para depurar un ejecutable
  gdb ./calculadora

Arreglos Unidimensionales y Multidimensionales

Unidimensional

tipoDeDato nombre[tamaño]

Multidimensional

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

Apuntadores

TipoDeDato *apuntador, variable;

apuntador = &variable;

Funciones

Sintaxis Básica para definir una función:

valorRetorno nombre (parámetros){

// bloque de código de la función

}

Lectura y escritura de datos

- fopen(): Abre una secuencia para que pueda ser utilizada y la asocia a un archivo.

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

  Modos de abrir los archivos

  - r: Abre un archivo de texto para lectura.
  - w: Crea un archivo de texto para escritura.
  - a: Abre un archivo de texto para añadir.
  - r+: Abre un archivo de texto para lectura / escritura.
  - w+: Crea un archivo de texto para lectura / escritura.
  - a+: Añade o crea un archivo de texto para lectura / escritura.
  - rb: Abre un archivo en modo lectura y binario.
  - wb: Crea un archivo en modo escritura y binario

- fclose(): Cierra una secuencia que fue abierta mediante una llamada a fopen().

  int fclose(FILE *apArch);

- fgets() y fputs(): Pueden leer y escribir, respectivamente, cadenas sobre los archivos.

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

- fprintf() y fscanf(): Se comportan exactamente como printf() (imprimir) y scanf() (leer), excepto que operan sobre archivo.

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

- 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).

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

ACORDEÓN DEL LENGUAJE DE PROGRAMACIÓN Ada

Aspectos importantes a tener en cuenta para usar Ada

- Ada no diferencia entre Mayúsculas y Minúsculas
- Para poder utilizar los elementos (subprogramas, tipos y variables) definidos en un paquete es necesario poner al principio la cláusula with.

Comentarios en Ada

- -- - Se utiliza dos guiones medios para comentar
- -- - Si se desea comentar varias líneas pones los dos guiones medios en todas las líneas donde se comento

Constantes y variables

Cada variable tiene asociado un identificador (un nombre) y un tipo de dato (entero, natural, carácter, etc.) que indica el tipo de información que puede guardar. Hay dos tipos de datos que podemos guardar: datos constantes y datos variables Para declarar constantes, justo antes de especificar el tipo de dato debemos utilizar la palabra reservada constant. Para declarar variable es parecido a las constante solo que esta vez no se coloca constant.

Formato para declarar constantes y variables

Nombre_Variable : Tipo_De_Dato;

Tipos de Datos Básicos

- Integer: Para cualquier número entero positivo o negativo.
- Natural: Más restrictivo que el anterior. Solamente permite el cero y los números positivos.
- Positive: Más restrictivo aún que el anterior. Solamente permite los números positivos.
- Float: Para almacenar cualquier número real en formato coma flotante. Los números en coma flotante debe tener como mínimo un digito antes y después del punto decimal.
- Character: Para almacenar una única letra (un carácter).
- String: Para almacenar una frase completa. Como Ada no sabe cuál es el tamaño máximo que queremos permitir, debemos decírselo en el momento de declarar la variable.
  Nombre_Completo : String (1 .. 40);

Identificadores

Palabras separadas por el carácter (_), cada palabra empieza con mayúscula y sigue en minúsculas.

Atributos

La forma de utilizarlos es poner un apóstrofe después del nombre del tipo y el nombre del atributo que queremos consultar.

- First, Last - Para saber cuál es el primer y el último valor de un determinado tipo de dato.
- Succ, Pred - Proporcionan el predecesor y sucesor de un determinado valor.
- Pos, Val - Nos dice la posición de un elemento de tipo enumerado en la declaración del tipo.
- Image, Value - Convierten desde y hacia String.
- tagged – Se ocupan en la declaración y extensión de tipos etiquetados.
- task - En la declaración y cuerpo de tareas y tipos tarea.
- terminate - Para terminar tareas.
- then - En la instrucción if-then-else y en el operador lógico de conjunción corto-circuitado (and then).
- type - En la declaración de tipos y en los parámetros de tipo en unidades genéricas.
- Use - Hace visibles directamente el contenido de un paquete
- while - Se utiliza para definir un tipo de bucle que se encuentra en la mayoría de los lenguajes de programación estructurados y cuyo propósito es repetir un bloque de código mientras una condición se mantenga verdadera.
- with - Se utiliza en los siguientes contextos: Clausula with, Orientación a objetos, Subprogramas y paquetes como parámetros de genéricos

Operadores Básicos

Operadores Aritmético

- + - Suma
- - - Resta
- * - Multiplicación
- / - División

Operadores relacionales

- < - Menor que
- > - Mayor que
- = - Igual a
- <= - Menor o igual que
- >= - Mayor o igual que
- /= - no igual a

Operadores lógicos

- and (y)
- or (o)
- xor (o exclusivo)
- not (negación)

Operador concatenación

- &
- Image

Entrada/Salida

En el primer nivel, el nivel llamado Text_IO, están los procedimientos encargados de leer una única letra (un carácter) y los procedimientos encargados de leer y escribir frases completas.

En el segundo nivel hay tres partes: una llamada Integer_IO, que contiene los procedimientos para leer y escribir números enteros, otra llamada Float_IO, para

leer y escribir números en coma flotante y otra llamada Enumeration_IO, para leer y escribir valores de tipos enumerados.

Escritura en pantalla (ejemplo Hola Mundo)

```
with Ada.Text_IO;

procedure Hola_Mundo is

begin

   Ada.Text_IO.Put_Line("¡Hola, mundo!");

end Hola_Mundo;
```

Letras

Para escribir y leer una letra utilizaremos directamente el primer nivel de Text_IO.

Escribir una letra

Para escribir una única letra utilizaremos Text_IO.Put.

Leer una Letra

Para leer una letra debemos declarar una variable de tipo Character para que Ada guarde la letra leida.

Escribir una frase

- Text_IO.Put(). Escribe en pantalla una frase. Cuando termina de escribir deja el cursor justo a continuación de la frase escrita.
  Si queremos que el cursor de escritura salte al principio de la línea siguiente tendremos que utilizar Text_IO.New_Line;.
- Text_IO.Put_Line(). Escribe en pantalla una frase. Al finalizar deja el cursor al principio de la línea siguiente a la escrita (por tanto es equivalente a llamar a Text_IO.Put y justo a continuación llamar a Text_IO.New_Line)

Leer una frase

- Text_IO.Get(). Lee del teclado una frase hasta que se llene la variable especificada
- Text_IO.Get_Line(). Lee del teclado una frase hasta que pulsemos la tecla RETURN o la tecla ENTER

Operadores para formaciones unidimensionales

El operador & concatena dos formaciones siempre que sean del mismo tipo (incluyendo dos String). También permite concatenar un elemento con una formación.

Atributos First, Last y Length

- First y Last - Para saber cuál es su primer y último índice válido

- Range - Es una abreviación del rango First .. Last. Se suele utilizar para recorrer todos los elementos de una formación.

Estructuras

Estructuras de control

- If
- Case
- If else
- elsif

Estructuras repetitivas

- Loop
- While
- For

En esta actividad considere que, al ser un nuevo Lenguaje para mí, debía integrar todo lo que debería saber para un funcionamiento correcto, no llegue a encontrar información en una sola pagina por lo cual consulte varias paginas donde compare la información y fui complementando mi acordeón para el lenguaje Ada Una de mis dificultades fue que la información era escaza, pero las que llegue a encontrar eran un completas.

Referencias:

- Francisco Guerra, J. M. (2002, 3 octubre). Fundamentos de Programación con Ada. clases-micros-para-com. Recuperado el 26 de febrero del 2021 de: http://www.iuma.ulpgc.es/~nunez/clases-micros-para-com/verilog/libro_ada.pdf

- Programación en Ada. (s. f.). Programación en Ada. Recuperado el 26 de febrero del 2021, de: https://recursos.mec.edu.py/kiwix/wikibooks_es_all_maxi/A/Programación_en_Ada/Texto_completo

- Facultad de ingeniería - UNAM. (2018, 6 abril). Manual de prácticas del laboratorio de Fundamentos de programación. Recuperado el 26 de febrero del 2021, de: http://odin.fi-b.unam.mx/salac/practicasFP/MADO-17_FP.pdf

- Introducción al lenguaje de programación Ada. (2019, septiembre). Departamento de Teoría de la Señal y Comunicación y Sistemas Telemáticos y Computación. Recuperado el 27 de febrero del 2021 de: https://www.cartagena99.com/recursos/alumnos/apuntes/0-intro-ada.pdf