Universidad Nacional Autónoma de México

Facultad de Ingeniería

Ingeniería en computación (110)

Estructuras de Datos y Algoritmos I

Actividad 1 viernes: Acordeón de lenguajes de programación

Martínez Miranda Juan Carlos

(26/02/2021)

Acordeón del lenguaje C

Sentencias de control

En un programa las instrucciones escritas en cada línea de código se ejecutan secuencialmente, sin embargo, no siempre es conveniente este orden de ejecución y para ello existen sentencias que permiten controlar la ejecución de instrucciones que nosotros como programadores definamos.

If – else

Esta sentencia trabaja bajo una condición que el programador establece, si se cumple la condición, se realizan las series de instrucciones definidas, en caso de no cumplirse la condición, se ignoran las líneas de código correspondientes al caso positivo y se ejecutan las que corresponden al caso contrario.

```
if (condición) {

(Serie de instrucciones a ejecutar en caso de que se cumpla la condición

} else {

(Instrucciones a ejecutar en caso de que no se cumpla la condición)

}
```

Switch Case

La sentencia switch funciona con base en el valor de una variable ya sea numérico o de tipo carácter, dependiendo del valor de esta variable se ejecutarán distintos casos de los cuales cada uno tiene instrucciones asignadas para su ejecución

```
switch (variable) {
        case 1: instrucciones para el primer caso
                break;
        case 2: instrucciones para el segundo caso
                break;
```

```
    case n: instrucciones para el enésimo caso
        break;

}
```

Ciclo for

Esta sentencia está diseñada para ejecutar varias veces una instrucción, para usar esta sentencia es necesario conocer el número de veces que se repetirán las instrucciones ya que de otra manera no es posible usarla correctamente.

```
for (inicialización de contador; condición; incremento/decremento) {
    Serie de instrucciones
}
```

Ciclo while

El ciclo while realizará una serie de instrucciones mientras se cumpla una condición, cuando deje de cumplirse dicha condición, ignorará las instrucciones, terminará y se proseguirá con el resto del código, para esta sentencia no es necesario conocer el número de veces que se ejecutará el ciclo, sin embargo, es necesario que exista un cambio dentro del ciclo para evitar un bucle infinito.

```
while (condición) {
    Serie de instrucciones y un cambio
}
```

Arreglos

Los arreglos son conjuntos de datos de un mismo tipo (entero, flotante, carácter…) los cuales están referenciados bajo un mismo nombre y son almacenados consecutivamente en localidades de memoria.

La sintaxis para declararlo es:

tipo nombre[tamaño]; ejemplo → int Arreglo [42];

Los arreglos multidimensionales son empleados para la representación de tablas, de manera que se utilizan dos subíndices, uno que corresponde a las filas y otro a las columnas.

La sintaxis para declararlo es:

tipo nombre [tamaño 1] [tamaño 2] [tamaño n]; ejemplo →int Arreglo [5][6][7];

Apuntadores

Un apuntador o puntero es una variable la cual contiene la dirección de memoria correspondiente a un dato o a la variable que contiene a dicho dato.

El operador & (Dirección) devuelve la dirección de una variable.

El operador * (Indirección) devuelve la dirección y el valor que contiene esa dirección.

}

Sintaxis para declarar apuntadores:

tipo_de_dato *nombre_del_apuntador; ejemplo → int *apuntador;

Funciones

Las funciones definidas por el programador son de mucha utilidad en un programa, pues en ellas se puede ejecutar una tarea en específico que puede ser requerida en más de un punto del código y manejar funciones hace que no se deban escribir todas las líneas de código correspondientes y sólo se haga una llamada a la función

y que esta retorne el resultado deseado, lo que ayuda al trabajo del programador y a la estética del código principal.

Para declarar una función, primero debe declararse lo que se conoce como prototipo, antes de la función principal, de igual manera, después del código principal, se escribe la función con toda la serie de instrucciones a ejecutar al momento de llamar a la función desde el código principal.

La sintaxis para declarar una función es:

```
tipo_de_retorno nombre (lista_de_parámetros);

main() {

 llamada;

}

tipo_de_retorno nombre (lista_de_parámetros){

serie_de_instrucciones;

return resultado;

}
```

Estructuras

Las estructuras son una colección de variables de distintos tipos de datos las cuales están relacionadas y no son referenciadas bajo el mismo nombre, el programador define el nombre de cada elemento que compone a la estructura.

La sintaxis de una estructura es:

```
struct nombre_estructura {
        tipo_de_dato nombre_variable;
        tipo_de_dato nombre_variable;
        tipo_de_dato nombre_variable;
```

}

Archivos

El uso de archivos al programar es bastante útil para almacenar datos y resultados obtenidos para acceder a ellos incluso después de la ejecución, así como modificarlos posteriormente.

Para abrir un archivo secuencial se establece un área de buffer en memoria principal, en donde la información se almacena temporalmente mientras se transfiere de memoria principal a secundaria.

Es necesario declarar un apuntador de tipo FILE, para abrir el archivo se utiliza la función fopen y asignar el resultado de la función al apuntador.

FILE *ap

ap= fopen(nombre_archivo, modo_apertura);

Las siguientes tablas muestran las funciones que pueden ser empleadas en el uso de archivos y los modos de apertura de estos.

Nombre	Función
fopen()	Abre un archivo.
fclose()	Cierra un archivo.
fgets()	Lee una cadena de un archivo.
fputs()	Escribe una cadena en un archivo
fseek()	Busca un byte especifico de un archivo.
fprintf()	Escribe una salida con formato en el archivo.
fscanf()	Lee una entrada con formato desde el archivo.
feof()	Devuelve cierto si se llega al final del archivo.
ferror()	Devuelve cierto si se produce un error.
rewind()	Coloca el localizador de posición del archivo al principio del mismo.
remove()	Borra un archivo.
fflush()	Vacía un archivo.

Modo	Significado
r	Abre un archivo de texto para lectura.
w	Crea un archivo de texto para escritura.
a	Abre un archivo de texto para añadir.
rb	Abre un archivo binario para lectura.
wb	Crea un archivo binario para escritura.
ab	Abre un archivo binario 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.
r+b	Abre un archivo binario para lectura / escritura.
w+b	Crea un archivo binario para lectura / escritura.
a+b	Añade o crea un archivo binario para lectura / escritura.

Acordeón de lenguaje Java

Clases

Las clases son plantillas las cuales definen la forma de un objeto, en ellas se definen las variables que se utilizarán y se conocen como atributos, así como los métodos los cuales se encargan de realizar una tarea.

Cada clase tiene sus propios atributos y métodos, por lo que no existen atributos globales, la única manera de compartir atributos y métodos de una clase a otra es mediante la herencia a una subclase y que esta además tenga sus propios atributos y métodos, sin embargo, no existe la herencia a múltiples subclases.

Modificadores de acceso a la clase

Los modificadores son los encargados de encapsular y controlar el acceso a los datos que conforman a un objeto, esto con el fin de aumentar la seguridad de los datos restringiendo el acceso a diferentes atributos y métodos.

public: Cualquier clase o subclase en el mismo paquete o en otro distinto tiene acceso a sus datos.

protected: Tienen acceso a sus datos las clases que están en el mismo paquete y subclases de otros, pero no clases de otros paquetes.

default: Sólo tienen acceso las clases que se encuentran en el mismo paquete.

private: Únicamente su propia clase tiene acceso a sus datos.

Constructores

Un constructor es el encargado de inicializar a un objeto al crearse, deben tener el mismo nombre que la clase y su sintaxis es similar a la de un método, sin embargo, los constructores no tienen un tipo de devolución explícito.

FUENTES DE CONSULTA

http://www.lcc.uma.es/~fvn/LabProg1/tema2_2004_2005.pdf

https://www.mheducation.es/bcv/guide/capitulo/8448148681.pdf

https://w3.ual.es/~abecerra/ID/archivos.pdf

https://www.dsi.fceia.unr.edu.ar/downloads/informatica/info_II/El_Lenguaje_C.pdf

https://developer.ibm.com/es/languages/java/tutorials/j-introtojava1/