**El lenguaje de programacion C:**

**Prefacio**:

C es un lenguaje de Proposito General que ofrece como ventaja economica de expresion, control de flujo y estructuras de datos modernas y un rico conjunto de operadores.

**Introduccion:**

C proporciona una variedad de tipos de datos. Los tipos fundamentales son caracteres, enteros y numeros de punto flotante de varios tamaños.

Las funciones pueden regresar valores de tipos basicos, estructuras, uniones o apuntadores. Cualquier funcion puede ser llamada recursivamente.

C mantiene la filosofia basica de que los programadores saben lo que estan haciendo; solo requiere que establezcan sus intenciones en forma explicita.

**Capitulo 1:** Introduccion general

1.1: Comencemos

Un programa en C, cualquiera que sea su tamaño, consta de funiones y variables. Una funcion contiene proposiciones que especifican las operaciones de calculo que se van a realizar, y las variables almacenan los valores utilizados durante los calculos.

La primera linea del programa: #include<stdio.h> indica al compilador que debe incluir informacion acerca de la biblioteca estandar de entrada/salida.

Un metodo para comunicar datos entre las funciones es que la funcion que llama proporciona una lista de valores, llamada argumentos a la funcion que esta invocando. Los parentesis que estan despues del nombre de la funcion encierran a la lista de argumentos. Main esta definido para ser una funcion que no espera argumentos, lo cual esta indicando por la lista vacia ().

Las proposiciones de una función están encerradas entre llaves {}.

Printf es una funcion de biblioteca que escribe la salida, o una secuencia de caracteres entre comillas se le llama cadena de caracteres o constante de cadena.

La secuencia \n en la cadena representa el carácter nueva linea; y hace avanzar a la siguiente linea.

1.2: Variables y expresiones aritmeticas

En C, se deben declarar todas las variables todas las variables antes de su uso, generalmente al principio de la funcion y antes de cualquier proposicion ejecutable. Una declaracion notifica las propiedades de una variable; consta de un nombre de tipo y una lista de variables.

El ciclo While funciona de la siguiente manera: se prueba la condicion entre parentesis. De ser verdadera se ejecuta. Luego la condicion se prueba nuevamente, y si es verdadera, el cuerpo se ejecuta de nuevo. Cuando la prueba resulta falsa la iteracion termina, la ejecucion continua en la proposicion que sigue al ciclo.

Printf es una funcion de proposito general. Su primer argumento es una cadena de caracteres que seran impresos, con cada % indicado en donde uno de los otros argumentos va a ser sustituido, y en que forma sera impreso. Ejemplo: %d especifica un argumento entero. Deben corresponder apropiadamente en numero y tipo, o se tendran soluciones incorrectas.

Printf no es parte del lenguaje C, es solo una util funcion de la Biblioteca estandar.

Si un operador numerico tiene un operando de punto flotante y otro entero, este ultimo sera covertido a punto flotante antes de hacer la operación.

1.3: La proposicion For

Es un ciclo, una forma generalizada del While. Dentro de los parentesis existen 3 secciones, separadaspor punto y coma. La primera, la inicializacion se ejecuta una vez, antes de entrar al ciclo. La segunda seccion es la condicion o prueba que controla el ciclo, esta condicion se evalua; si es verdadera se ejecuta, despues el incremento de avance se ejecuta y la condicion se vuelve a evaluar. El ciclo termina si la condicion se hace falsa. La incializacion, la condicion y el incremento pueden ser cualquier expresion.

1.4: Constantes Simbolicas

Una linea #Define define un nombre simbolico o constante simbolica como una cadena de caracteres especial: #Define nombre texto de reemplazo. A partir de esto, cualquier ocurrencia del nombre se sustituira por el texto de reemplazo correspondiente. El nombre tiene la misma forma que el nombre de una variable: una secuencia de letras y digitos que comienza con una letra. El texto de reemplazo puede ser cualquier secuencia de caracteres; no esta limitado a numeros.

#include<stdio.h>

#define LOWER 0

#define UPPER 300

#define STEP 20

Las cantidades LOWER,UPPER y STEP son constantes simbolicas, no variables, por lo que no aparecen entre las declaraciones. Los nombres de constantes simbolicas se escriben con letras mayusculas, de modo que se pueda distingui facilmente de los nombres de variables escritos con minusculas. No hay punto y coma al final de una linea #define

1.5: Entrada y Salida de Caracteres:

La entrada y salida de texto, sin importar donde fue originada o hacia donde se dirige, se trata como flujos (Streams) de caracteres. Un flujo de texto es una secuencia de caracteres seguidos de un carácter nueva linea. La biblioteca es responsable de hacer que cada secuencia de entrada o salida este de acuerdo a este modelo.

La biblioteca estandar proporciona varias funciones para leer o escribir un carácter a la vez, getchar lee el siguiente carácter de entrada de una secuencia de texto y lo devuelve como su valor. Esto es, despues de

C = getchar()

La variable C contiene el carácter de entrada. Los caracteres provienen normalmente del teclado.

La funcion putchar escribe un carácter cada vez que se invoca:

Putchar(C)

Escribe el contenido de la variable entera C como un carácter, generalmente en pantalla.

1.5.1: Copia de Archivos:

Con getchar y putchar se puede escribir una cantidad sorprendente de codigo util sin saber nada mas de acerca de entrada y salida.

El operador de relacion != significa “no igual a “.

EOF es un entero definido <stdio.h>, pero el valor numerico especifico no importa mientras que no sea el mismo que ningun valor tipo char.

1.5.2: Conteo de Caracteres:

++ significa que incrementa en uno. Estos operadores pueden ser prefijos (++nc) como postfijos (nc++).

1.5.3: Conteo de Lineas:

Un carácter escrito entre apostrofos representa un valor entero igual al valor numerico del carácter en el conjunto de caracteres de la maquina. Esto se llama una constante carácter. Por ejemplo ‘A’ es una constante de carácter; en el conjunto ASCII de caracteres, su valor es 65. ‘\n’ es 10 en codigo ASCII.

El doble signo de igualdad == es la notacion de C para expresar “Igual a”.

1.5.4: Conteo de Palabras:

Cuenta las palabras usanado la definicion de que una palabra es cualquier secuencia de caracteres que no contiene espacio en blanco ni tabulacion ni nueva linea.

Cada vez que el programa encuentra el primer carácter de una palabra, contabiliza una palabra mas. La variable state registra si actualmente el programa esta o no sobre una palabra. Es preferible usar constantes simbolicas IN y OUT que los valores literales 1 y 0, porque hacen al progama mas legible.

La linea:

ni = nw = nc = 0

inicializa a las tres variables en cero. Las asignaciones se asocian de derecha a izquierda.

El operador || significa “O” y existe un correspondiente operador && para AND, su sentencia es mas alta que la de ||. Las expresiones conectadas por && o || se evaluan de izquierda a derecha.

El ejemplo muestra tambien un else, el cual especifica una accion alternativa si la condicion de una proposicion if es falsa. La forma general es:

If (expresion)

Proposicion1;

Else

Proposicion2;

Una y solo una de las 2 proposiciones asociadas con un if-else se realizan.

1.6: Arreglos:

La declaracion:

Int ndigit[10];

Declara ndigit como un arreglo de 10 enteros. En C los subindices de arreglos comienzan en cero, por lo que los elementos son ndigit[0], ndigit[1],.., ndigit[9]. Esto se refleja en los ciclos for que inicializan e imprimen el arreglo.

Por definicion, los char son solo pequeños enteros, por lo que las variables y las constantes char son identicos a los Int en expresiones aritmeticas.

El patron:

If (condicion)

Proposicion;

Else if (condicion)

Proposicion;

…

…

Else

Proposicion;

se encuentra frecuentemente en programas como una forma de expresar una decision multiple. Las condiciones se evaluan en orden desde el principio hasta que se satisface alguna Condicion; en ese punto se ejecuta la proposicion correspondiente, y al contruccion completa termino.

Si no se satisface ninguna de las condiciones, se ejecuta la proposicion que esta despues del else final, si esta existe.

1.7: Funciones:

En lenguaje C, una funcion es el equivalente a una subrutina o funcion en Fortran, o a un procedimiento o funcion en Pascal. Una funcion porporciona una forma conveniente de encapsular algunos calculos, que se puede emplear despues sin preocuparse de su implantacion.

Una definicion de funcion tiene la siguiente forma:

Tipo-de-retorno nombre-de-funcion(declaracion de los parametros, si los hay).

{

Declaraciones

Proposiciones

}

La funcion power se invoca dos veces por main, en la linea:

Printf(“%d%d%d\n”,i,power(2,i),power(-3,i));

Cada llamado para dos argumentos a power, que cada vez regresa un entero, al que se pone formato y se imprime.

Int power(int base,int n)

Declara los tipos y los nombres de los parametros, asi como el tipo de resultado que la funcion devuelve. Los nombres que emplean power para sus parametros son locales a la funcion y son invisibles a cualquier otra funcion: otras rutinas pueden utilizar los mismos nombres sin que exista problema alguno.

el valor que calcula power se regresa a main por medio de la proposicion return, a la cual le puede seguir cualquier expresion:

return expresion.

Una funcion no necesita regresar un valor; una proposicion return sin expresion hace que el control regrese al programa, pero no devuelve algun valor de utlidad.

La declaracion:

Int power(Int m,int n);

Precisamente antes de main, indica que power es una funcion que espera dos argumentos Int y regresa un Int. Esta declaracion, se le llama funcion prototipo, debe coincidir con la definicion y uso de power.

Los nombres de los parametros no necesitan coincidir; de hecho, son optativos en el prototipo de una funcion, de modo que para el prototipo se pudo haber escrito:

Int power(Int,Int);

No obstante, unos nombres bien seleccionados son una buena documentacion, por lo que se emplearan frecuentemente.

Los parametros se nombran entre parentesis y sus tipos de declaran antes de abrir la llave izquierda; los parametros que no se declaran se toman como Int.

1.8: Argumentos-llamadas por valor:

En C, todos los argumentos de una funcion se pasan “Por valor”. Significa que la funcion que se invoca recibe los valores de sus argumentos en variables temporales y no en las originales.

En C la funcion que se invoca no puede alterar directamente una variable de la funcion que hace la llamada; solo puede modificar su copia privada y temporal. Sin embargo, la llamada por valor es una ventaja, conduce a elaborar programas mas compactos con pocas variables extrañas, debido a que los parametros se tratan en la funcion invocada como variables locales convenientemente inicializadas. Ejemplo:

Int power(Int Base, Int n)

{

Int p;

For(p=1;n>0;--n)

P = p \* base;

Return p;

}

El parametro n se utiliza como una variable temporal, y se decrementa hasta que llega a cero; cualquier cosa que se le haga a n dentro de Power no tiene efecto sobre el argumento con el que se llamo originalmente a power.

Cuando sea necesario,es posible que una funcion modifique una variable dentro de la rutina invocada. La funcion que llama debe proporcionar la direccion de la variable que sera cambiada y la funcion que se invoca debe declarar que el parametro sea un apuntador y tenga acceso a la variable a traves de el.

La historia es diferente con los arreglos. Cuando el nombre de un arreglo se emplea como argumento, el valor que se pasa a la funcion es la localizacion o la direccion del principio del arreglo. Al colocarle subindices a este valor, la funcion puede tener acceso y alterar cualquier elemento del arreglo.

1.9: Arreglo de Caracteres:

El tipo de arreglo mas comun en C es el de caracteres. La funciones getline y copy estan declaradas al pirncipio del programa, que se supone esta contenido en un archivo.

Int getline(char s[],Int Lim);

Especifica que el primer argumento, S,es un arreglo, y el segundo, lim,es un entero. La longitud del arreglo S no es necesaria en getline, puesto que su tamaño se fija en main. En getline se utiliza return para regresar un valor a quien lo llamo. Esta linea declara que getline regresa un Int.

Algunas funciones regresan un valor util; otras, como copy, se emplean unicamente por su efecto y no regresan un valor. El tipo de retorno de copy es void, el cual establece explicitamente que ningun valor se regresa.

En getline se coloca el carácter ‘\0’ (carácter nulo, cuyo valor es cero) al final del arreglo que esta creando, para marcar el fin de la cadena de caracteres.

La especificacion de formato %s dentro de printf espera que el argumento correspondiente sea un cadena representada de este modo; copy tambien se basa en el hecho de que su argumento de entrada se termina con ‘\0’, y copia este carácter dentro del argumento de salida.

Para un usuario de getline no existe forma de saber con anticipacion cuan larga podra ser una linea de entrada, por lo que getline revisa un posible desbordamiento (overflow). Por otro lado, el usuario de copy ya conoce (o puede averiguar) cual es el tamaño de la cadena, por lo que decidimos no agregar comprobacion de errores en ella.

1.10: Variables externas y alcance:

Las variables que estan en main, son privadas o locales a ella. Debido a que son declaradas dentro del main, ninguna otra funcion puede tener acceso directo a ellas.

Cada variable local de una funcion comienza a existir solo cuando se llama a la funcion, y desaparece cuando la funcion termino.

Como una alternativa de las variables automaticas (variables locales), es posible definir variables que son externas a todas las funciones, esto es, variables a las que toda funcion puede tener acceso por su nombre. Debido a que es posible tener acceso global a las variables externas, pueden ser usadas en lugar de listas de argumentos para comunicar datos entre funciones. Como las variables externas se mantienen permanentemente en existencia, mantienen sus valores aun despues de que regresa la funcion que los fija.

Una variable externa debe definirse, exacytamente una vez, fuera de cualquier funcion; esto fija un espacio de almacenamiento para ella; esto establece el tipo de la variable. La declaracion debe ser una proposicion **extern** explicita, o bien puede ser implicita en el contexto.

**extern** Int max;

**extern** Char longest[];

Desde el punto de vista sintactico, las definiciones externas son exactamente como las definiciones locales, pero como ocurren fuera de las funciones, las variables son externas. Antes de que la funcion pueda usar una variable externa, se debe hacer saber el nombre de la variable a la funcion. Una forma de escribir una declaracion **extern** dentro de la funcion; la declaracion es la misma que antes, excepto por la palabra reservada **extern.**

Bajo ciertas sircustancias, la declaracion **extern** se puede omitir. Si la definicion de variable externa ocurre dentro del archivo fuente antes de su uso por una funcion en particular, entonces no es necesario el uso de una declaracion **extern** dentro de la funcion. Una practica comun, es poner las definiciones de todas las variables externas al principio del archivo fuente y despues omitir todas las declaraciones **extern**.

Si el programa esta compuesto de varios archivos fuente y una variable se define *archivo1* y se utiliza en *archivo2* y *archivo3*, entonces se necesita declaraciones **extern** en *archivo2* y *archivo3* para conectar las ocurrencias de la variable.

Empleamos cuidadosamente las palabras definicion y declaracion cuando nos referimos a variables externas. **Definicion** Se refiere al lugar donde se crea la variable o se le asigna un lugar de almacenamiento; **Declaracion** se refiere al lugar donde se establece la naturaleza de la variable pero no se le asigna espacio. Las variables externas existen siempre, aun cuando no hacen falta.

**Capitulo 2:** Tipos, operadores y expresiones

Las variables y las constantes son los objetos de datos basicos que se manipulan en un programa. Las declaraciones muestran las variables que se van a utilizar y establecen el tipo que tienen y algunas veces cuáles son sus valores iniciales. Los operadores especifican lo que se hará con las variables. Las expresiones combinan variables y constantes para producir nuevos valores.

2.1: Nombres de variables:

Existen algunas restricciones en los nombres de las variables y de las constantes simbolicas. Los nombres se componen de letras y dígitos; el primer carácter debe ser una letra. El carácter “Subrayado ( \_ )” cuenta como letra.

Es conveniente elegir nombres que estén relacionados con el proposito de la variable.

2.2: Tipos y tamaños de datos:

Hay unos cuantos tipos de datos básicos en C:

|  |  |
| --- | --- |
| Char | Un solo byte, capaz de contener un carácter del conjunto de caracteres local. |
| Int | Un entero, normalmente del tamaño natural de los enteros en la maquina en la que se ejecuta. |
| Float | Punto flotante de precision normal. |
| Double | Punto flotante de doble precision. |

El calificador *signed* o *unsigned* puede aplicarse a char o a cualquier entero. Los numeros *unsigned* son siempre positivos o cero y obedecen las leyes de la aritmetica modulo 2n donde n es el numero de bits en el tipo. Asi, por ejemplo, si los char son de 8bits, las variables *unsigned* char tienen valores entre 0 y 255, en tanto las variables *signed* char tienen valores entre -128 y 127.

Los archivos de encabezado *headers* estandas <limits.h> y <float.h> contienen constantes simbolicas para todos esos tamaños, junto con otras propiedades de la maquina y del compilador.

**Capitulo 4:** Funciones y la Estructura del Programa

4.3: Variables Externas

Un programa en C consta de un conjunto de objetos externos, que son variables o funciones. El adjetivo “externo” se emplea en contraste con “interno”, que recibe los argumentos y las variables definidas dentro de las funciones. Las variables externas se definen fuera de cualquier funcion, y por lo tanto, estan potencialmente disponibles para muchas funciones. Las funciones en si mismas son siempre externas, C no permite definir funciones dentro de otra. Las variables y funciones externas tiene la propiedad de que todas las referencias a ellas por el mismo nombre, incluso desde funciones compiladas separadamente, son referencias a la misma cosa. (El estandar llama a esta propiedad *ligado externo*).

Debido a que las variables externas son accesibles globalmente, proporcionan una alternativa a los argumentos de funciones y a los valores de retorno para comunicar datos entre funciones. Cualquier funcion puede tener acceso a variables externas.

Si un gran numero de variables se debe compartir entre funciones, las variables externas son mas convenientes y eficientes. Sin embargo, se debera aplicar con precaucion, pues puede tener un efecto negativo sobre la estructura del programa y dar lugar a programas con demasiadas conexiones de datos entre funciones.

Las varibles externas son utiles debido a su mayor alcance y tiempo de vida. Las variables automaticas son internas a una funcion y su existencia se inicia cuando se entra a la funcion y desaparecen cuando esta se abandona. Por otro lado, las variables externas son permanentes, de modo que retienen sus valores de la invocacion de una funcion a la siguiente.