Unidad 1

Un lenguaje de programación es un sistema notacional para describir computaciones en una forma legible tanto para la máquina como para el ser humano.

El diseño de lenguajes exige:

- legibilidad: por parte del ser humano es un requisito complejo y sutil.
- abstracción: permitir concentrarse en un problema al mismo nivel de generalización, dejando de lado los detalles irrelevantes.

Tipos de Abstracciones

- Abstracciones de datos: Técnica de inventar nuevos tipos de datos que sean más adecuados a una aplicación y, por consiguiente, facilitar la escritura del programa.
- Abstracciones de control: Resume propiedades de la transferencia de control, es decir, la modificación de la trayectoria de ejecución de un programa en una determinada situación.
- Abstracciones procedimentales: Es una secuencia nombrada de instrucciones que tienen una función específica y limitada.

Sintaxis: estudia las reglas de formación de frases. Las reglas de sintaxis nos dicen cómo se escriben los enunciados, declaraciones y otras construcciones del lenguaje.

Semántica: hace referencia al significado de esas construcciones.

Traductor: Es un "procesador de lenguajes" que acepta programas en cierto lenguaje fuente como entrada y produce programas funcionalmente equivalentes en otro lenguaje objeto.

Simulación por software o Interpretación: es un software que recibe un programa en lenguaje de alto nivel, lo analiza y lo ejecuta. Para analizar el programa completo, va traduciendo sentencias de código y ejecutándolas si están bien, así hasta completar el programa origen.

Compilador: Un compilador es un traductor que transforma un programa escrito en un lenguaje de alto nivel (lenguaje fuente) en otro programa escrito en un lenguaje de bajo nivel.

Conversión explícita o forzada: conjunto de funciones integradas que el programador puede llamar para realizar una conversión de tipos.

Eficiencia

- Eficiencia de código: un diseño del lenguaje que permite que el traductor genere un código ejecutable eficiente. Ej. uso de declaraciones anticipadas de variables (estáticas).
- Eficiencia de traducción: un diseño del lenguaje que permite que el código fuente se traduzca con rapidez y con un traductor de tamaño razonable.
- Eficiencia de implementación: eficiencia con la que se puede escribir un traductor.
- Eficiencia de la programación: facilidad para escribir programas en el lenguaje.

Regularidad

La regularidad indica lo bien que se integran las características del lenguaje. Es una cualidad que implica que haya pocas restricciones en el uso de sus constructores particulares y en la forma en la que se comportan las características del lenguaje.

- Ortogonalidad: Se refiere al atributo de combinar varias características de un lenguaje de manera que la combinación tenga significado.
- Uniformidad: Hace referencia a la consistencia de la apariencia y comportamiento de los constructores del lenguaje.

Paradigma de programación

Un paradigma de programación "consiste en un método para llevar a cabo cómputos y la forma en la que deben estructurarse y organizarse las tareas que debe realizar un programa".

Se dividen en 2, declarativa e imperativa:

Programación declarativa: Está basada en describir el problema declarando propiedades y reglas que deben cumplirse, en lugar de instrucciones. La solución es obtenida mediante mecanismos internos de control, sin especificar exactamente cómo encontrarla. No existen asignaciones destructivas, y las variables son utilizadas con transparencia referencial. Uso de mecanismos matemáticos para optimizar el rendimiento de los programas.

Programación imperativa: se basa en dar instrucciones al ordenador de cómo hacer las cosas en forma de algoritmos, en lugar de describir el problema o la solución.

Lenguajes imperativos

Un lenguaje procedural o imperativo es un lenguaje controlado por mandatos o instrucciones. El proceso de ejecución de programas procedurales por la computadora, tiene lugar a través de una serie de estados, cada uno definido por el contenido de la memoria, los registros internos y los almacenes externos durante la ejecución.

Unidad 2

Datos Simples

Objetos de datos: Representación de un elemento u objeto de la realidad de manera que pueda ser procesado por una computadora

Tiempo de vida: Tiempo durante el cual el objeto puede usarse para guardar valores de datos.

Enlace o ligadura de un elemento de programa: Elección de una propiedad dentro de un conjunto de propiedades posibles.

Tiempo de enlace: Momento del programa durante el cual se realiza el enlace.

Tipos de ligadura:

- Los tipos de datos se fija en tiempo de definición del lenguaje
- L os posibles valores se determinan en tiempo de implementación del lenguaje
- El tipo de dato asociado es un enlace realizado en tiempo de traducción
- Si se trata de un tipo definido por el programador (lo admite Pascal o C), el conjunto de sus posibles valores se fija en tiempo de compilación.

Tabla de símbolos: Es una estructura de datos que contiene una entrada por cada identificador declarado encontrado en el programa fuente, con la siguiente información: tipo de variable tipo de parámetro formal nombre del subprograma entorno de referencia, etc.

Tipo de dato: es una clase de objetos de datos ligados a un conjunto de operaciones para crearlos y manipularlos.

Especificación:

- Atributos: distinguen objetos de ese tipo.
- Valor: especifica valores que pueden tomar objetos de este tipo de dato.
- Operaciones: refieren a distintas formas de manipular los objetos de ese tipo.

Implementación:

- Representación de almacenamiento: refiere a la forma que se representan en memoria.
- Representación de algoritmos y procedimientos que definen las operaciones.

Declaraciones: es un enunciado del programador que sirve para comunicar al traductor información que permite establecer ligaduras

Ventajas:

- Informan al traductor acerca de la representación de almacenamiento para un objeto de datos.
- Permiten que el traductor, determine cual es la operación particular a la que hace referencia el operador, en el caso de operaciones polimórficas.
- Informan sobre el tiempo de vida de un objeto.
- Verificación estática de Tipos

Verificación de tipos: es el proceso que realiza el compilador o el intérprete para verificar que todas las construcciones en un programa tengan sentido.

Verificación estática: se realiza durante la traducción.

Verifica la validez de los argumentos y determina el tipo de datos del resultado, información que guarda para verificar operaciones posteriores.

Si la operación es homonimia o polimórfica, el nombre de la operación se puede reemplazar por el nombre de la operación específica que usan esos argumentos.

Como no se realizan marcas de tipo en los objetos en tiempo de ejecución, se gana almacenamiento y tiempo de ejecución.

Verificación dinámica: se realiza durante la ejecución.

Para implementarla se guarda una marca de tipo en cada objeto (descriptor), que indica el tipo de datos asociado al mismo. Cuando se realiza la verificación, lo primero que se efectúa es la verificación de marcas de tipo de argumentos.

La operación se realiza si los tipos de argumentos son correctos, de lo contrario se indica un error.

Para cada operación, se anexan las marcas de tipo a sus resultados, para verificación de las operaciones siguientes.

Operación de asignación:

Rvalue: hace referencia al valor que contiene una variable

Lvalue: se refiere a la localidad de la variable

Tipo de dato: booleano

Representa a objetos de datos que pueden tomar uno de dos valores posibles: True o False.

Implementación:

Dos maneras de representación dentro de la unidad de almacenamiento direccionables (un byte o una palabra):

1- Se usa un solo bit del byte para representar con 0 o 1 y los restantes se ignoran.
2- Un valor 0 en toda la unidad de almacenamiento representa falso, cualquier otro valor representa verdadero.

Tipo de dato: carácter

Especificación: toma como valor un solo carácter. El conjunto de posibles valores se corresponde con el conjunto de caracteres que maneja el hardware y el Sistema Operativo subyacente.

Implementación: Objetos de datos de tipo carácter manejados por el hardware y el S.O subyacente, debido a su uso en la entrada y salida de información.

Tipo de dato: Puntero

Un objeto del tipo apuntador, posee la dirección o localidad de otro objeto (valor L del objeto) o puede contener un apuntador nulo.

Permite:

- Al programador, en tiempo de ejecución crear variables y destruirlas cuando las considere innecesarias
- Crear estructuras de datos dinámicas.
- Devolver resultados a través de los parámetros de una función.

Tipos de datos estructurados

Especificación:

Estructuras de tamaño fijo: número de componentes invariable durante su tiempo de vida.

Estructura de tamaño variable: número de componentes cambia durante su tiempo de vida. Usan generalmente un apuntador para vincular componentes.

Organización de las componentes:

- Unidimensional o según una serie lineal (arreglo unidimensional, registro, cadena de caracteres, listas.)
- Multidimensional (arreglo multidimensional, registro cuyos componentes son registros, listas cuyos componentes son listas, etc.)

Representación de almacenamiento

- Representación Secuencial: usa un solo bloque de memoria contigua, incluye tanto componentes como descriptor.
- Representación Vinculada: usa varios bloques de memoria no contiguos vinculados a través de punteros.

Arreglo unidimensional

Estructura homogénea de tamaño fijo organizada como una serie lineal simple.

Arreglos bidimensionales

En general, un arreglo de cualquier dimensión tiene representación en orden por filas.

Esta representación implica que el arreglo se divide en subarreglos para cada elemento del intervalo del primer subíndice. Cada uno de estos subarreglos es un arreglo.

También existe una estrecha relación entre indexación en un arreglo y aritmética de punteros, donde esto b[i] es equivalente a esto *(b+i).

Cadenas de caracteres

En C se utiliza la librería string.h, una cadena de caracteres fija es asi: char nom []=”FIN”, mientras que una variable es asi: char nom[20].

Registros

Estructura de datos compuesta por un número fijo de componentes de igual o distinto tipo, de longitud fija.

Implementación: Se usa una representación secuencial de almacenamiento. La selección de una componente se implementa con facilidad pues los nombres de los campos se conocen en tiempo de traducción.

Unidad 3

Definición de tipos

Permite crear nuevos tipos de datos y operaciones sobre ese tipo, de modo que pueda ser usado como un tipo de dato provisto por el lenguaje. Una definición de tipos proporciona un nombre de tipo junto con una declaración que describe la estructura de una clase de objetos de datos. El nombre del tipo se convierte en el nombre de esa clase de objetos de datos, y cuando se necesita trabajar con un objeto de datos particular basta con proporcionar el nombre del tipo en lugar de repetir la descripción completa de la estructura de datos.

Sistema de tipos

Un sistema de tipos incluye los métodos utilizados para la construcción de tipos, el algoritmo de equivalencia de tipos, y las reglas de inferencia y corrección de tipos. Si un lenguaje presenta un sistema de tipos completo que pueda aplicarse estáticamente y donde todos los errores de corrupción se detecten lo antes posible, entonces se dice que el lenguaje es fuertemente tipificado.

- Ventajas de la definición de tipos:
- Simplificación la estructura del programa
- Modificación más eficiente.
- En el uso de subprogramas, facilita el pasaje de argumentos, pues evita repetir la descripción del tipo de datos.

Definición, invocación y activación de subprogramas

Definición de un subprograma: es una propiedad estática de lenguaje; en tiempo de traducción es la única información disponible. La traducción de la definición de un subprograma es una plantilla que permite generar activaciones en tiempo de ejecución, asi mismo la plantilla se divide en 2: segmento de código y registro de activación.

El segmento de código es la parte estática compuesta por las constantes y el código ejecutable generado a partir de enunciados del cuerpo de la función.

Prólogo: bloque de código que el traductor introduce al comienzo del segmento de código.

Epílogo: conjunto de instrucciones que el traductor inserta al final del bloque de código ejecutable

Activación de un subprograma: se genera en tiempo de ejecución cuando se lo llama o invoca. Al terminar la ejecución, la activación se destruye.

El registro de activación es la parte dinámica, compuesta por: Parámetros, Resultados de la función, Datos locales, Punto de retorno, Áreas temporales de almacenamiento, Vinculaciones para referencias de variables no locales.

Funciones en C

Una función es un conjunto de sentencias que realiza una determinada tarea, que retorna como resultado cero o un valor.

El uso de funciones definidas por el programador permite dividir un programa en un cierto número de componentes más pequeñas, cada una de éstas con un propósito específico y determinado, logrando así programas más fáciles de codificar y depurar.

Pasaje de parámetros

Por valor

Los parámetros reales se evalúan al momento de la llamada a la función y se transforman en los valores que toman los parámetros formales durante la ejecución de la función. Los parámetros formales pueden cambiar sus valores a través de asignaciones sin que estos cambios afecten los valores de los parámetros reales.

Desventaja: se produce duplicación del área de memoria.

Por dirección

Consiste en el paso por valor de una dirección, por ello puede usarse para cambiar el contenido de la memoria apuntada por ese puntero. El pasaje de parámetros por dirección permite retornar más de un resultado desde la función.

Por referencia

Para realizar un pasaje por referencia el parámetro actual a pasar debe ser una variable con una dirección asignada. El pasaje por referencia pasa la ubicación de la variable, por lo que el parámetro formal se transforma en un alias del parámetro actual de modo que cualquier cambio que se realiza en el parámetro formal se siente en el parámetro actual.

Clasificación de variables

Por su tipo: int, float, char, etc.

Por su almacenamiento: globales, estáticas, automáticas y dinámicas.

Variables automáticas:

Son las declaradas en la lista de parámetros o en el cuerpo de una función, se almacenan en el registro de activación de la función Su alcance restringido a la función en la que se declaran

Variables externas:

Se definen una sola vez, fuera de cualquier función y pueden ser inicializadas, su alcance no se restringe a una función, todas las funciones pueden tener acceso a ella a través de su nombre y se almacenan en el área de almacenamiento estático.

Variables estáticas:

Son locales a una función, se declaran con el especificador “static”, pueden ser inicializadas, su alcance se restringe a la función en la que se declaran y se almacenan en el área de almacenamiento estático, por ello mantienen la información a lo largo de la ejecución.

Bloque en lenguaje C

Un bloque es una secuencia de declaraciones, seguidas por una secuencia de enunciados, rodeados por marcadores sintácticos.

Tabla de símbolos

La tabla de símbolos es una estructura de datos, se crea en tiempo de traducción, permite dar apoyo a la inserción, búsqueda y cancelación de identificadores con sus atributos asociados

Los símbolos se guardan en la tabla con su nombre y una serie de atributos opcionales que dependen del lenguaje y objetivos del procesador. Entre ellos se encuentra:

- Nombre de identificador.
- Dirección a partir de la cual se almacenará la variable en tiempo de ejecución, dirección a partir de la cual se colocará el código en caso de funciones.
- Tipo del identificador. sí es una función, el tipo del resultado.
- Tipo de los parámetros de las funciones o procedimientos

Unidad 4

Recursividad

Es una alternativa a la iteración, un proceso mediante el cual se puede definir una función en términos de sí misma.

Al construir una función recursiva, se debe asegurar la existencia de una estructura de selección que permita identificar:

1. Caso base (puede haber más de uno): permite detener la invocación sucesiva de la función (no la vuelve a invocar).
2. Caso general (puede haber más de uno): permite que la función se invoque a sí misma con valores de parámetros que cambian en cada llamada; acercándose cada vez más al caso base.

En la ejecución, las sentencias que aparecen después de cada invocación no se resuelven inmediatamente, éstas quedan pendientes.

Cada invocación recursiva, genera un registro de activación y se apila en el stack o pila. Esos registros de activación se van desapilando en el orden inverso a como fueron apilados

Recursividad vs Iteración

Ambas se basan en una estructura de control:

- la iteración utiliza una estructura de repetición
- la recursión utiliza una estructura de selección.

Ambas implican repetición:

- la iteración utiliza la repetición de manera explícita
- a recursión consigue la repetición mediante repetidas llamadas a una misma función.

Ambas involucran una prueba de terminación:

- la iteración termina cuando falla la condición de continuación del ciclo
- la recursión termina cuando se reconoce un caso base.

Unidad 5

Estructura de datos dinámicas

Pueden crearse en tiempo de ejecución y su espacio de almacenamiento pueda liberarse cuando no se las necesite y modificar su tamaño durante la ejecución del programa.

Estáticas o de Almacenamiento Fijo:

- El sistema asigna el área que ocupará, a partir de su declaración (identificador y tipo)
- El almacenamiento permanece fijo durante la ejecución de la función que la declara
- No se pueden utilizar esas posiciones de memoria para otros fines durante la ejecución

Dinámicas:

- Se crean en tiempo de ejecución, en cualquier punto, no sólo al entrar a un subprograma, a petición
- Se puede liberar su espacio de almacenamiento cuando no se las necesite
- En el caso de estructuras, se puede modificar su tamaño durante la ejecución del programa.

Montículo

Operaciones (Puede ser explicita o implícita):

- Alojamiento: Se demanda un bloque para almacenar un objeto de cierto tamaño. Resultado de la función es un puntero a un trozo contiguo de memoria dentro del montículo
- Desalojo: Se indica que ya no es necesario conservar un objeto, la memoria que ocupa queda libre para ser reutilizada

Características del lenguaje para definir variables dinámicas

- Poseer un tipo elemental de datos apuntador, llamado también tipo de referencia o de acceso. (puntero) Una operación de creación de objetos de tamaño fijo, que devuelve el valor L del bloque de almacenamiento creado y ese valor L se convierte en el valor r de una variable del tipo apuntador.
- Una operación de desreferenciar para objetos del tipo apuntador, que permite acceder al valor del objeto al cual apunta.

Características e implementación de punteros

Direcciones absolutas: El apuntador contiene la dirección de memoria real del bloque de almacenamiento del objeto.

Direcciones relativas: El apuntador contiene el desplazamiento del objeto respecto a la dirección base de un bloque de almacenamiento más grande.

Manejo del Montículo en lenguaje C

El Sistema Operativo maneja una tabla de direcciones de memoria que indican el espacio de memoria ocupado (para que un programa no modifique direcciones de memoria que no le corresponden, corrompiendo la memoria de otros procesos / programas /información).

Funciones malloc y free:

- **Función Malloc (Memory ALLOCation: asignación de memoria):** Solicita a sistema operativo un bloque de memoria en el montículo, del tamaño especificado en el argumento, el sistema operativo registra en su tabla de direcciones de memoria que un bloque ha sido ocupado, y envía a malloc la dirección de inicio de ese bloque. Resultado: dirección de comienzo del bloque asignado o la constante NULL, si no hay espacio disponible. El tipo del bloque asignado es Void. (se le debe hacer un cast al tipo usado)
  Formato: malloc (tamaño del bloque)
- **Función free:** indica que el bloque de memoria asignado a través de mallloc se ha dejado de usar, con esta información el Sistema Operativo registra en su tabla de direcciones de memoria que dicho bloque ha sido desocupado. La variable puntero conserva el valor que tenía antes de free, es decir no cambia su contenido, por este motivo, para evitar errores de acceso, resulta ser una buena práctica inicializar la variable después de su liberación con la constante NULL.
  Formato: free(variable puntero);

Algunos Inconvenientes que pueden surgir:

- Presencia de elemento basura: cuando habría espacio disponible para nuevo uso, pero dicho espacio no está en la lista de espacios libres, por ello se ha vuelto inaccesible.
- Referencias desactivadas: Esto sucede cuando al finalizar el tiempo de vida de un objeto, el almacenamiento se recupera para ser reasignado, pero no se destruyen todas las rutas de acceso.

Listas

Una lista es un conjunto de elementos del mismo tipo que puede crecer o contraerse sin restricciones. Todos los elementos son accesibles y se puede insertar y suprimir un elemento en cualquier posición de la lista.

Las listas se pueden implementar meditan arreglos, denominadas como listas secuenciales como punteros, denominadas listas enlazadas.

Características de las listas secuenciales:

- Se debe conocer cantidad de elementos o asignar suficiente espacio de memoria, si la cantidad no se conoce a priori.
- Los elementos se almacenan en celdas contiguas de memoria.
- Sus elementos se acceden de manera directa Insertar un elemento en cualquier lugar de la lista, obliga al desplazamiento de una posición a todos los elementos que le siguen.
- La eliminación de un elemento, excepto el último, requiere desplazamientos para llenar el vacío formado.

Listas enlazadas

- Cada elemento deberá contener además de su información intrínseca, un puntero a otro elemento de estructura similar.

- El campo enlace apunta al siguiente nodo de la lista excepto el último nodo que contiene NULL, para indicar el final de la lista. Este valor de apuntador NULL no hace referencia a un lugar de memoria, sino que denota que el apuntador apunta a nada.

Características de las listas enlazadas

En esta representación cada elemento (nodo) de una lista debe tener dos partes:

- Una contiene la información de cualquier tipo de dato simple o estructurado (excepto File)
- La otra parte, contiene un enlace o puntero que indica la posición del siguiente elemento.

Cada elemento tiene la estructura de tipo struct Los elementos se pueden almacenar en celdas de memoria no contiguas

Ventaja: Desaparece el problema de los costosos desplazamientos de elementos para insertar y eliminar componentes de la lista en cualquier posición y no se necesita conocer a priori la cantidad de elementos de la lista.

Desventajas: Búsquedas Secuenciales y requiere un espacio adicional para los apuntadores.

Unidad 6

Archivos

Estructura compuesta por datos almacenados en forma organizada en un dispositivo de almacenamiento secundario: discos rígidos, CD, pendrive, DVD

Ventajas del uso de archivos:

- Datos pueden ser usados por distintos programas y en distintos procesos.
- Capacidad de almacenamiento de memoria secundaria superior a memoria principal
- El uso de archivos permite tener en memoria principal sólo aquella parte de datos que necesita ser accedida por el programa en un momento dado
- El tamaño de los archivos está limitado sólo por la cantidad de memoria secundaria disponible.

El acceso a los datos de un archivo es más lento que los datos residentes en memoria principal tiempos mecánicos (orden de los milisegundos) Vs tiempos electrónicos (orden nanosegundos).

Organización de Archivos

- Organización Secuencial: las componentes se ubican en posiciones físicas contiguas en el soporte de almacenamiento
- Organización Directa: las componentes ocupan posiciones físicas relacionadas mediante una clave, que puede ser un dato de la componente o el resultado de un cálculo.

Criterios de clasificación de archivos

Por la dirección del flujo de datos:

- Archivos de entrada
- Archivos de salida.
- Archivos de entrada / salida.

Según el tipo de valores permitidos a cada byte:

- Archivos de texto
- Archivos Binarios

Según el tipo de acceso (técnica utilizada para leer o grabar las componentes):

- Secuencial
- Directa o aleatoria

Según la longitud de registro:

- Archivos de longitud variable
- Archivos de longitud constante

Archivos organizados secuencialmente con acceso secuencial

FILE: estructura de dato (struct) definida por C en la biblioteca "stdio.h".

Con esta declaración, en tiempo de compilación se habilita un espacio de memoria para una estructura de tipo FILE, que queda referenciada a través de la variable declarada.

La definición de esta estructura depende del compilador, pero en general mantiene información sobre:

- la posición actual de lectura/escritura
- el estado del archivo
- un puntero a un buffer, área de memoria que sirve de almacenamiento intermedio entre el archivo físico y el programa que lo maneja.

A medida que se realizan operaciones sobre el archivo la estructura se actualiza.

En realidad, una variable de tipo FILE * representa un flujo de datos que se asocia con un dispositivo físico de entrada/salida (el archivo “real” estará almacenado en disco).

Estructura FILE

```
typedef struct {
  short level;                 // nivel de ocupación del buffer
  unsigned flag;               //  indicadores de control
  char fd;                     // descriptor del archivo
  char hold;                   //  carácter de ungetc()
  short bsize;                 //  tamaño del buffer
  unsigned char *buffer;       // puntero al buffer
  unsigned char *curp;         // posicion en curso
  short token;                 // se emplea para control
        }FILE;                 // tipo definido con el nombre FILE
```

Las componentes se encuentran ubicadas en posiciones físicas contiguas en soporte de almacenamiento y pueden accederse: secuencialmente o en forma directa

Características de los archivos secuenciales:

- Escritura de nuevos datos al final del archivo.
- Lectura de una componente concreta en forma secuencial desde el lugar donde está posicionado el puntero. Necesidad de posicionarse al comienzo del archivo.
- Sólo se pueden abrir para lectura o escritura, nunca de los dos modos a la vez.

Se dividen en: Archivos de caracteres, de texto y binarios.

Archivos de caracteres

Son una secuencia de caracteres almacenados en un dispositivo de almacenamiento secundario e identificado por un nombre de archivo.

Archivo de texto

Un archivo de texto contiene un conjunto de “líneas” de caracteres de longitud variable y están separadas por un salto de línea.

Archivo sin formato

Estos archivos son tratados como un bloque de datos binarios (en bytes) que tan solo tienen significado si se conoce la estructura: distribución y tipos de datos. Pueden ser de acceso secuencial o directo.

Archivos organizados secuencialmente con acceso directo

El archivo tiene en las componente un elemento llamado clave, que está en relación con la posición física del almacenamiento. Esto permite la ubicación precisa de la componente en el archivo. Cuando se accede en forma directa a un archivo de acceso directo, hay que tener en cuenta algunas características:

- La escritura de nuevos datos se realiza al final del archivo sin que se pierda la relación de la clave con la posición física de la componente.
- Para leer una componente concreta se utiliza la clave para ubicar dicha componente.