Clase 4- unidades de programa

miércoles, 17 de abril de 2024 13:12

Unidades -> abstracción, acción abstracta

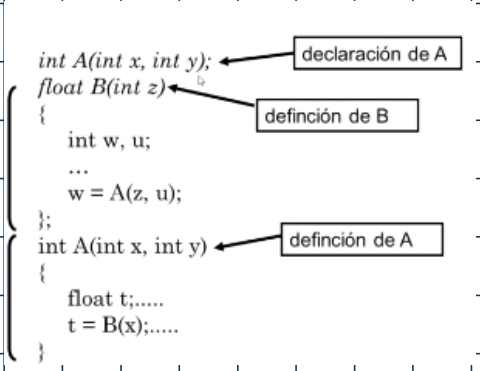
Unidedas con esquema de return-> rutina

Tienen nombre, alcance, tipo, lvalue, rvalue

Nombre: para llamar rutina tengo que conocer su nombre, tiene que estar en mi alcance

Alcance: rango de instrucciones donde se conoce. En general desde donde se define para abajo.

Declaración: encabezado. Cuerpo + declaración--> definición
Se puede extender alcance poniendo declaración arriba



C por defecto asume portiotipo
Sino b no podría llamar a a pq está abajo.
Pascal usa forward para extender alcance.

Tipo: de parámetros y de valor de retorno. Llamado a rutina es correcto si cumple con eso.

L-valor: dlnde se almacenan las sentencias de la rutina (cuerpo)

r-valor: lugar donde se invoca a la rutina. Se dice que es estático (en general), pq el llamado está en una parte específica y eso se ve al comilar. Pero a veces es dinámico

```c
int main()
{
    int y;
    void (*punteroAFuncion)();

    printf("Probando R-VALUE funciones\n");

    /* Pureba de Llamada a función R-VALUE estático*/
    y= 0;
    uno(y);

    /* Pureba de Llamada a función R-VALUE dinámico*/
    y= 1;
    punteroAFuncion = &uno;
    punteroAFuncion(y);

    return 0;
}
```

estático

dinámico

Comunicación e/rutinas:

- Accecso a ambiente no local: no muy bueno pq tengo que buscar a quién pertenece y demás. Toco siempre el mismo dato
- Parámetros: especifico qué datos quiero compartir.
    - Formal: definir la rutina
    - Reales: invocar rutina

En ejecución
Segmento de código/programa: instrucciones, se almacenan en la memoria de instrucción. FIJO. ZONA C

Registro de activación/datos: datos locales de la memoria. CAMBIANTE. ZONA D

PROCESADOR ABSTRACTO - SIMPLESEM

Memoria de Código: C(y) valor almacenado en la yésima celda de la memoria de código. Comienza en cero

Memoria de Datos: D(y) valor almacenado en la yésima celda de la memoria de datos. Comienza en cero *y* representa el l-valor, *D(y)* o *C(y)* su r-valor

Ip: puntero a la instrucción que se esta ejecutando.

- Se inicializa en cero en cada ejecución se actualiza cuando se ejecuta cada instrucción.
- Direcciones de C

Ejecución:

- obtener la instrucción actual para ser ejecutada (C[ip])
- incrementar ip
- ejecutar la instrucción actual

LENGUAJES ESTÁTICOS:

- Cargan todo antes de empezar a ejecutar y queda hasta el final de la ejecución
- No puede haber recursión
- Espacio necesario para ejecutar se deduce del código.
- Todas las variables son estáticas.

BASADO EN PILA:

- Forma una pila en zona de datos
- Solo arranca ejecución con lo que necesita, a medida que se necesitan más rutinas se las va alocando y se va desalocando lo que se termina.
- La memoria a usar es predecible, sigue lifo

DINÁMICO:

- Datos se alocan dinamicamente
- No se puede modelizar con pila pq el programador puede hacer objetos de dato en cualquier momento durante ejecución
- No se puede predecir tamaño
- Datos se alocan en heap.