Pilas

Representación en memoria estática y dinámica de las Pilas

Memoria estática

Las técnicas de asignación de memoria estática son sencillas. La asignación de memoria puede hacerse en tiempo de compilación y los objetos están vigentes desde que comienza la ejecución del programa hasta que termina. En los lenguajes que permiten la existencia de subprogramas, y siempre que todos los objetos de estos subprogramas puedan almacenarse estáticamente se aloja en la memoria estática un registro de activación correspondiente a cada uno de los subprogramas. Estos registros de activación contendrán las variables locales, parámetros formales y valor devuelto por la función.

Consideraciones:

- Error en tiempo de ejecución de índice fuera del rango.
- Se debe conocer con anticipación el tamaño de la estructura.
- Se guardan en memorias adyacentes.

Ventajas:

- La velocidad de acceso es alta.
- Lógica simple.
- Son más fáciles de diseñar.

Desventajas:

- No se puede modificar el tamaño de la estructura en tiempo de ejecución.
- No es óptimo con grandes cantidades de datos.
- Desperdicio de memoria cuando no se utiliza en su totalidad del tamaño.
- Menor capacidad, debido a que cada celda de almacenamiento requiere más transistores.
- Mayor costo por bit.
- Mayor consumo de Potencia

Morgado Jacome Eduardo #17211545

Memoria dinámica

La memoria dinámica es un espacio de almacenamiento que se solicita en tiempo de ejecución. De esa manera, a medida que el proceso va necesitando espacio para más líneas, va solicitando más memoria al sistema operativo para guardarlas.

El medio para manejar la memoria que otorga el sistema operativo, es el puntero, puesto que no podemos saber en tiempo de compilación dónde nos dará huecos el sistema operativo (en la memoria de nuestro PC). Un dato importante es que como tal este tipo de datos se crean y se destruyen mientras se ejecuta el programa y por lo tanto la estructura de datos se va dimensionando de forma precisa a los requerimientos del programa, evitándonos así perder datos o desperdiciar memoria si hubiéramos tratado de definirla cantidad de memoria a utilizar en el momento de compilar el programa.

Cuando se crea un programa en el que es necesario manejar memoria dinámica el sistema operativo divide el programa en cuatro partes que son: texto, datos (estáticos), pila y una zona libre o heap. En el momento de la ejecución habrá tanto partes libres como partes asignadas al proceso por lo cual si no se liberan las partes utilizadas de la memoria y que han quedado inservibles es posible que se “agote” esta parte y por lo tanto la fuente de la memoria dinámica. También la pila cambia su tamaño dinámicamente, pero esto no depende del programador sino del sistema operativo.

Ventajas:

- Es posible disponer de un espacio de memoria arbitrario que dependa de información dinámica (disponible sólo en ejecución): Toda esa memoria que maneja es implementada por el programador cuando fuese necesario.
- Otra ventaja de la memoria dinámica es que se puede ir incrementando durante la ejecución del programa. Esto permite, por ejemplo, trabajar con arreglos dinámicos.

Desventajas

- Es difícil de implementar en el desarrollo de un programa o aplicación.
- Es difícil implementar estructuras de datos complejas como son los tipos recursivos. Por ello necesitamos una forma para solicitar y liberar memoria para nuevas variables que puedan ser necesarias durante la ejecución de nuestros programas: Heap.

Morgado Jacome Eduardo #17211545

- La memoria dinámica puede afectar el rendimiento. Puesto que con la memoria estática el tamaño de las variables se conoce en tiempo de compilación, esta información está incluida en el código objeto generado. Cuando se reserva memoria de manera dinámica, se tienen que llevar a cabo varias tareas, como buscar un bloque de memoria libre y almacenar la posición y tamaño de la memoria asignada, de manera que pueda ser liberada más adelante. Todo esto representa una carga adicional, aunque esto depende de la implementación y hay técnicas para reducir su impacto.

Operaciones básicas con pilas

Una pila cuenta con 2 operaciones imprescindibles: apilar y desapilar, a las que en las implementaciones modernas de las pilas se suelen añadir más de uso habitual. Cada vez que se realiza una operación push() la pila aumenta de tamaño y el tope se modifica siendo ahora el elemento añadido. En el caso de la operación pop() disminuye el tamaño de la pila, y el tope será el elemento debajo del elemento a ser removido.

- *Crear*: se crea la pila vacía. (constructor)
- *Tamaño*: regresa el número de elementos de la pila. (size)
- *Apilar*: se añade un elemento a la pila.(push)
- *Desapilar*: se elimina el elemento frontal de la pila.(pop)
- *Cima*: devuelve el elemento que está en la cima de la pila. (top o peek)
- *Vacía*: devuelve cierto si la pila está vacía o falso en caso contrario (empty)

Ejemplo de push() y pop() en Javascript

pop(): Remueve el ítem al final de la pila.

```
let cats = ['Bob', 'Willy', 'Mini'];
cats.pop(); // ['Bob', 'Willy']
```

Morgado Jacome Eduardo #17211545

push(): Agrega el ítem al final de la pila.

```
let cats = ['Bob'];
cats.push('Willy'); // ['Bob', 'Willy']
cats.push('Puff', 'George'); // ['Bob', 'Willy', 'Puff', 'George']
```

Bibliografía

Sin especificar. (2004). Unidad III – Estructuras Lineales., del Instituto Tecnológico de Piedras Negras, Sitio web: http://itpn.mx/recursosisc/3semestre/estructuradedatos/Unidad%20III.pdf

Morgado Jacome Eduardo #17211545