Programación II

Memoria Dinámica

Memoria

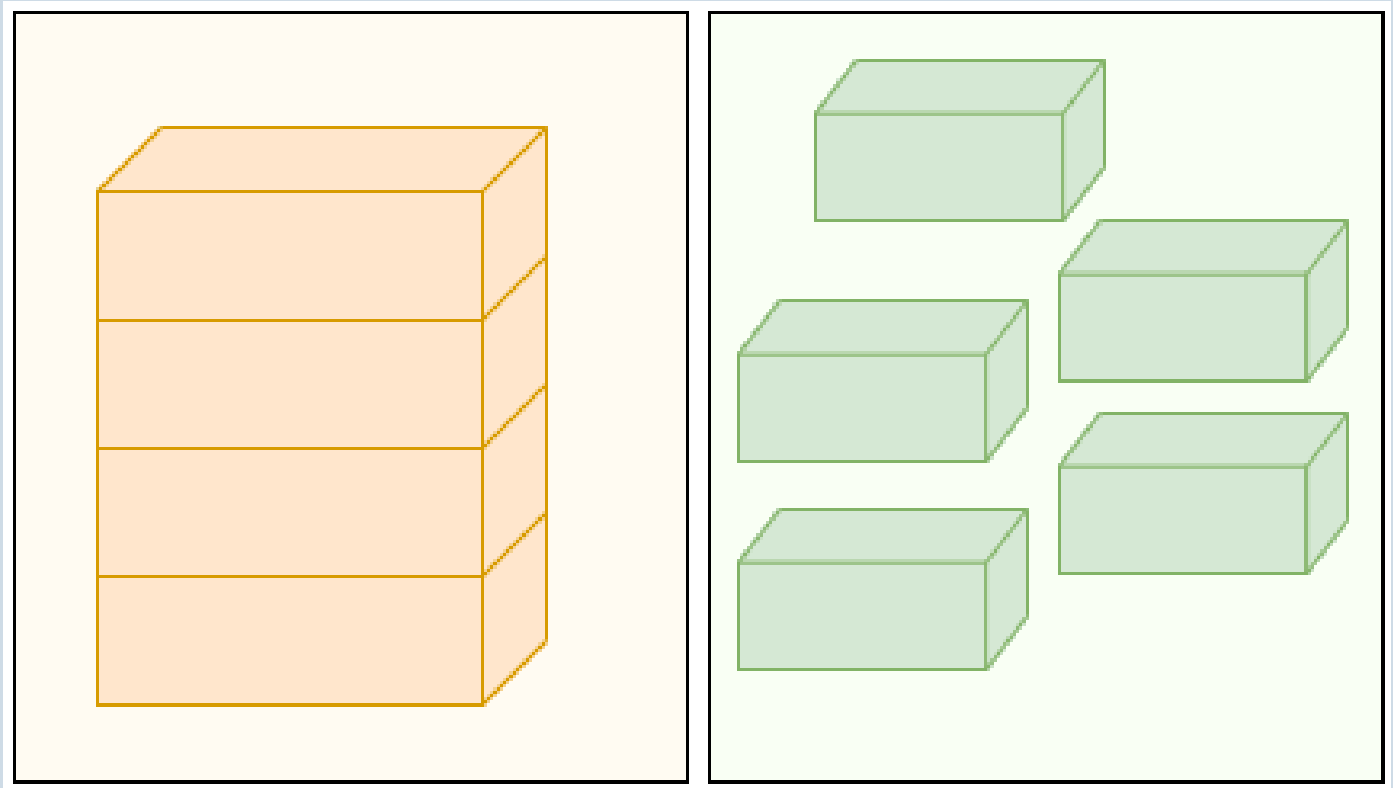
Hasta ahora, hemos trabajado con variables “***automáticas***”.

¿Qué significa esto?

Significa que declaramos variables en alguna parte de nuestro código (en el **main** o en alguna **función**) y esa variable existe desde el principio al final de las llaves **{}** en las cuales está declarada.

Memoria

Cuando ejecutas un programa, el sistema operativo asigna **memoria RAM** para que ese programa funcione. Esa memoria se divide principalmente en dos zonas importantes:



Memoria Stack

- Es una zona de memoria **privada** y **limitada** (en Windows típicamente 1 MB).
- Se utiliza para almacenar:
  Variables locales (por ejemplo: int x = 5;)
  Datos de funciones (como parámetros, variables locales, etc.)
- Esta memoria es **rápida** de acceder.
- La memoria del stack la gestiona el compilador: se reserva y se libera automáticamente.

Memoria Heap

- Es una zona de memoria **grande** y **compartida**, gestionada por el sistema operativo.

Usamos la memoria heap cuando necesitamos:

- Más memoria que la memoria stack admite.
- Cuando sepamos la cantidad de memoria a necesitar durante el tiempo de ejecución del programa.
- Cuando queremos que una variable pueda utilizarse más allá de su ámbito (sin declararla de manera global)

Memoria

- Una variable puede figurar en la memoria stack o en la memoria heap dependiendo de cómo trabajemos con ella.

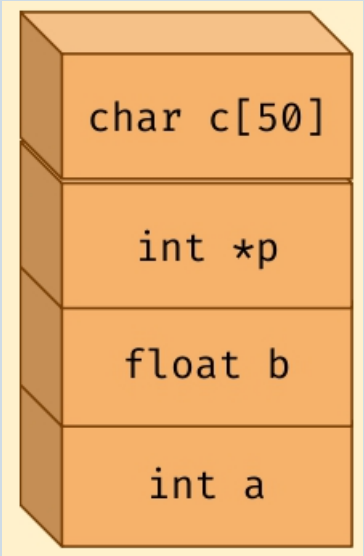
- Hasta el momento siempre utilizamos la memoria stack. La memoria dinámica permitirá ubicar nuestras variables en la memoria heap.

- La memoria heap es una memoria compartida por varios programas ejecutándose en el sistema operativo. No hay garantía de poder obtener la necesaria para nuestro programa.

Memoria Stack

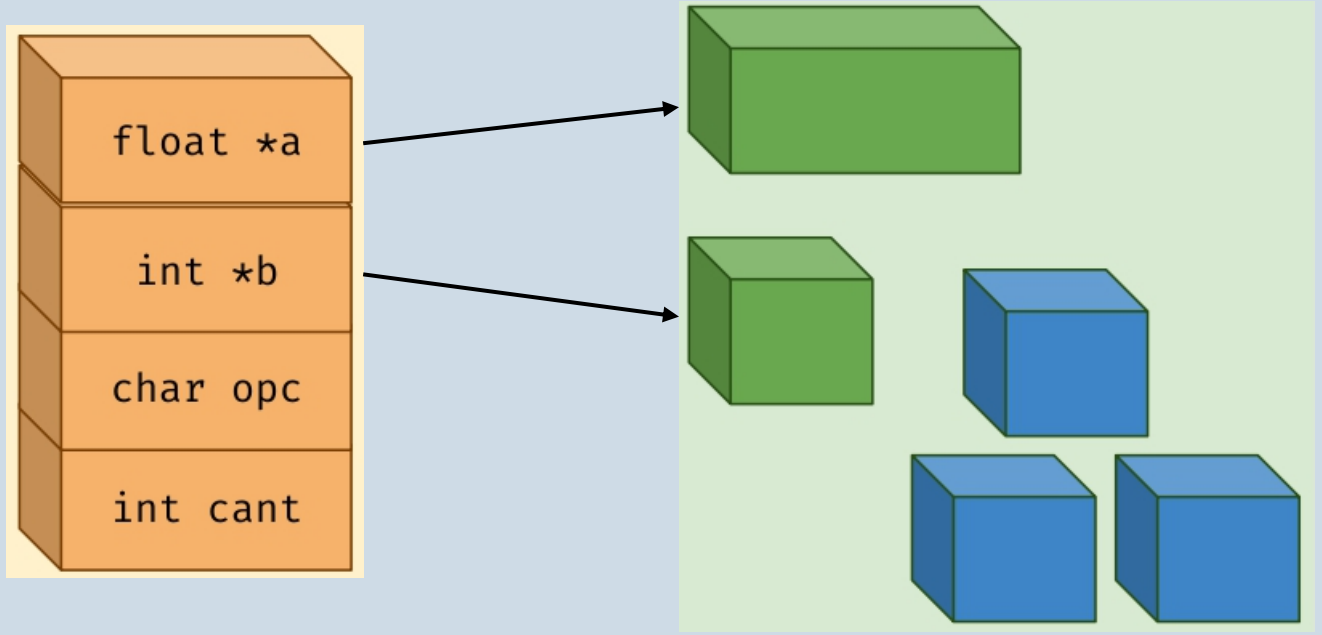
- Cada variable que declaremos en una función (incluso main) se ubica en la memoria stack.
- La memoria stack es limitada. De superar su limite genera una excepción (desbordamiento de pila o stack overflow).

```
int main(){
  int a, float b;
  int *p;
  char c[50];
}
```



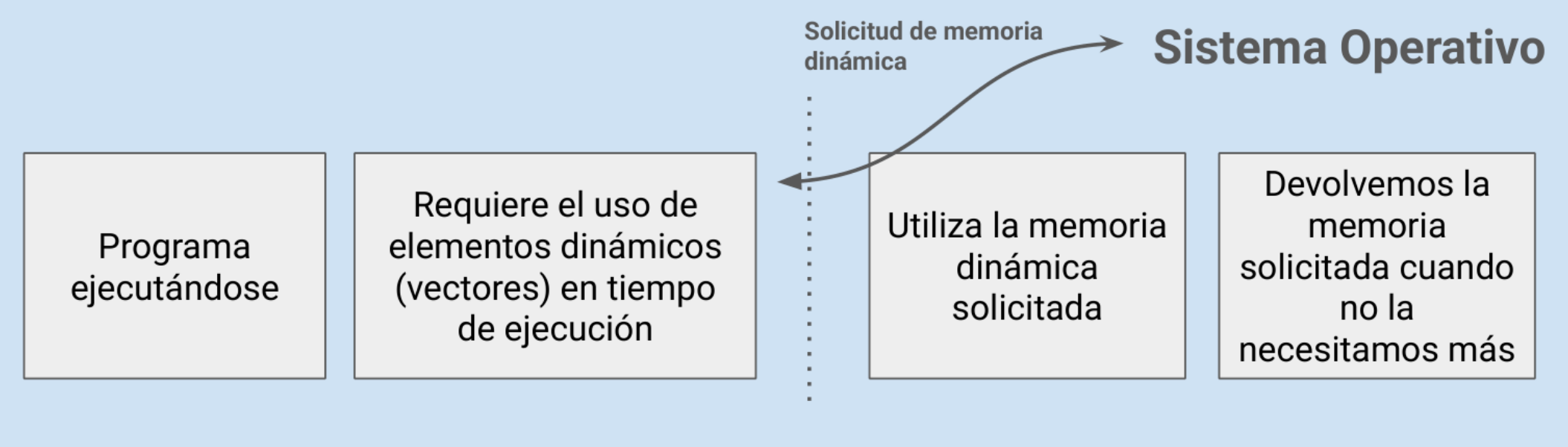
Memoria Heap

- Se crea un puntero en la memoria stack que, luego de pedir memoria, apunta al comienzo del espacio de memoria solicitado.



Asignación dinámica de memoria

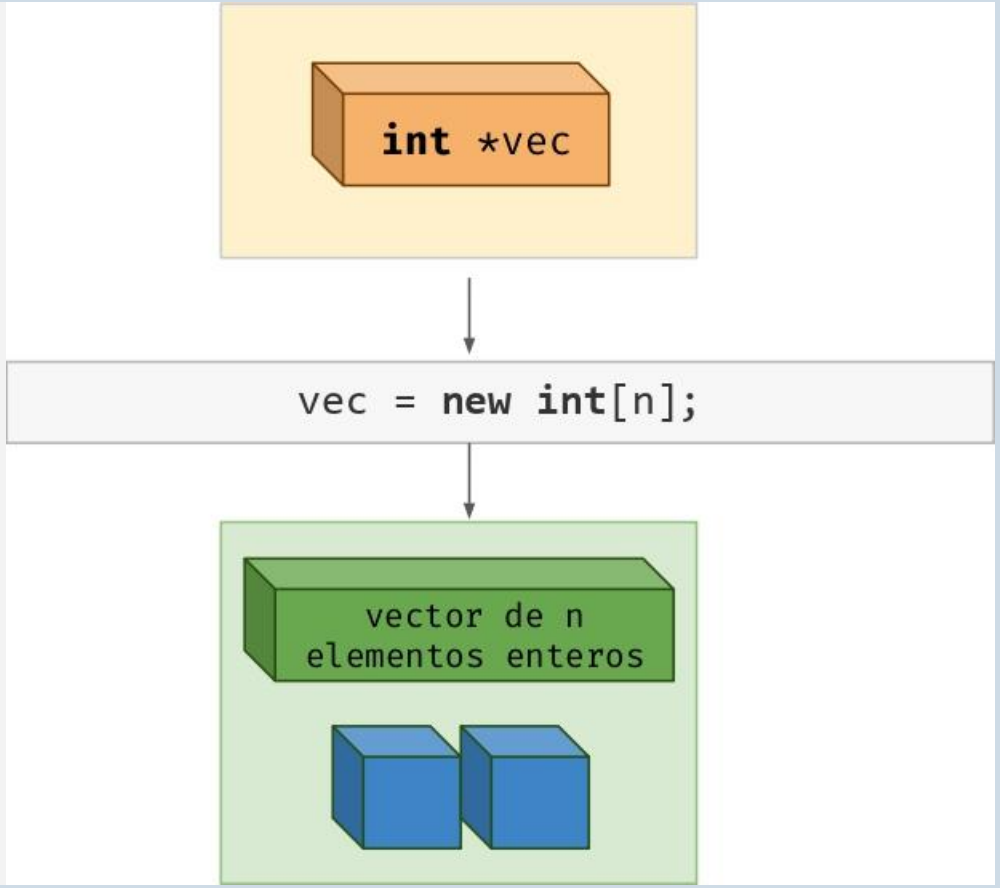
Proceso que permite solicitar memoria adicional al sistema operativo en tiempo de ejecución.



Operador new

Operador utilizado para pedir memoria dinámica sobre un puntero previamente declarado.

```
#include <iostream>
using namespace std;
int main(){
  /* Memoria dinámica para
  un vector de N elementos */
  int *vec, n;
  cin >> n;
  vec = new int[n];
  if (vec == NULL)
    return 1; // No hay memoria
  // Resto del programa
  delete []vec; // Liberación de la memoria
  return 0;
}
```



Operador delete

Operador utilizado para liberar memoria dinámica sobre un puntero previamente utilizado

```
delete []vec; // Liberación de la memoria
```

Créditos

- Angel Simón, docente UTN FRGP