Laboratorio de Computación II

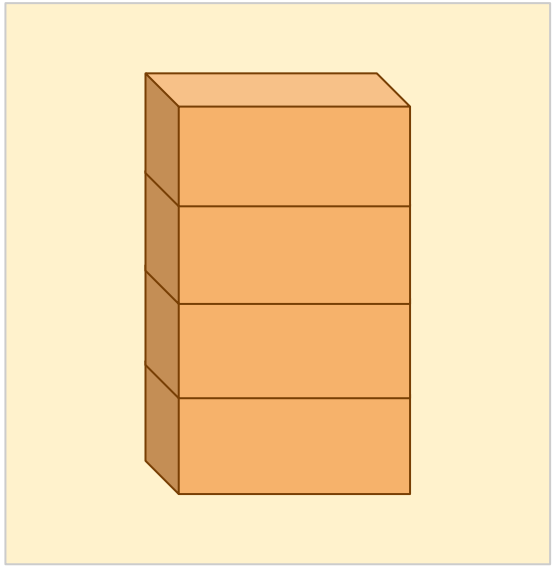
Asignación dinámica de memoria

Asignación dinámica de memoria

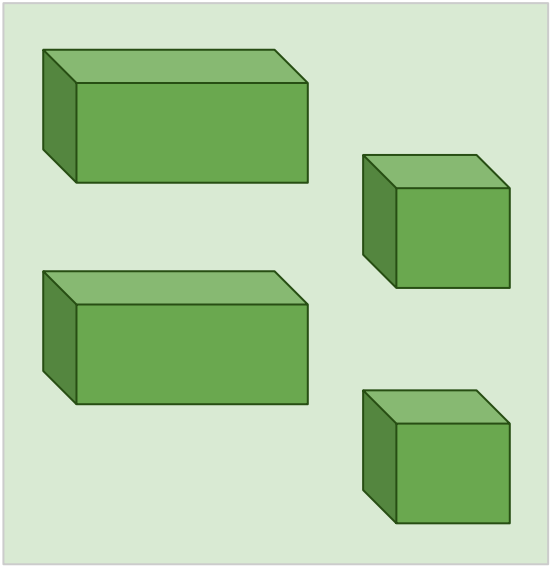
- Proceso que permite solicitar memoria adicional al sistema operativo en tiempo de ejecución.
- Nos permite utilizar la memoria exacta que necesitamos para trabajar y, una vez utilizada, debemos liberarla.
- Nos permite utilizar una mayor cantidad de memoria que de la manera convencional.

Memoria

La memoria se puede clasificar en stack o heap según su ubicación.



Memoria stack

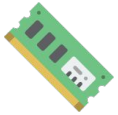


Memoria heap

Memoria

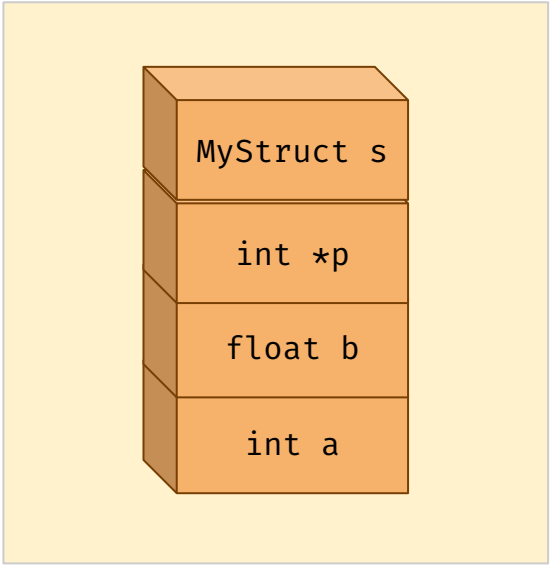
- Una variable puede figurar en la memoria stack o en la memoria heap dependiendo de cómo la declaremos.
- Hasta el momento siempre utilizamos la memoria stack. La memoria dinámica permitirá ubicar nuestras variables en la memoria heap.
- La memoria heap es una memoria compartida por varios programas ejecutándose en el sistema operativo. No hay garantía de poder obtener la necesaria para nuestro programa.

Memoria stack



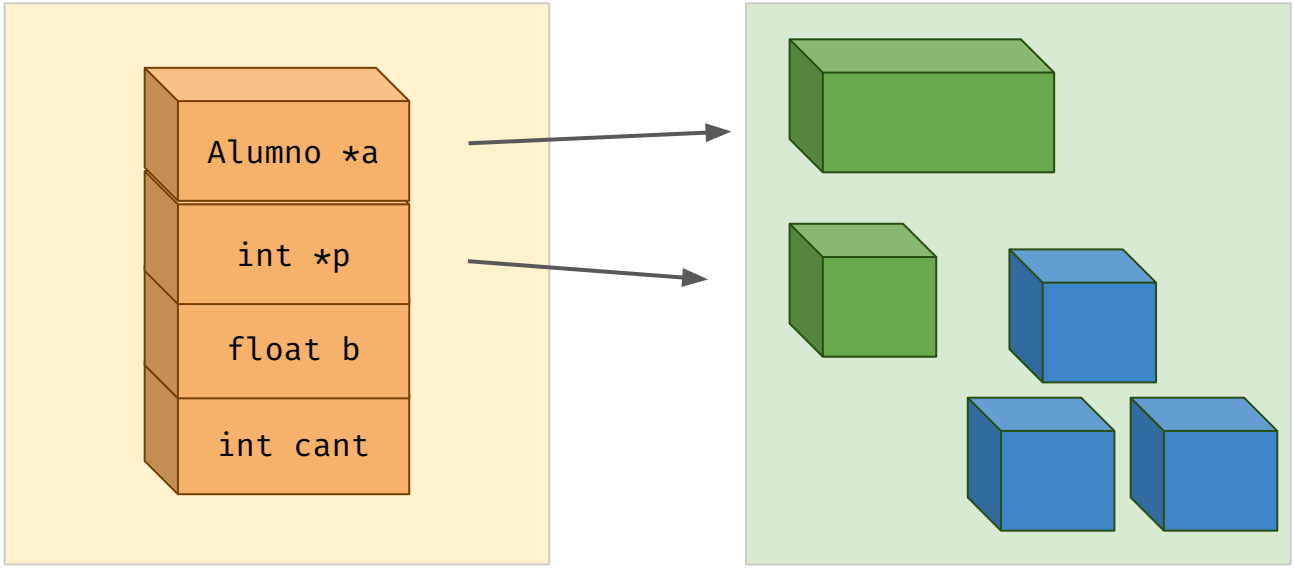
- Cada variable que declaremos en una función (incluso main) se ubica en la memoria stack.
- La memoria stack es limitada. De superar su límite genera una excepción (desbordamiento de pila o stack overflow).

```
int main(){
  int a, float b;
  int *p;
  struct MyStruct s;
}
```



Memoria heap

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



malloc

Función de C utilizada para pedir memoria dinámica

```
void * malloc (long bytes);
```

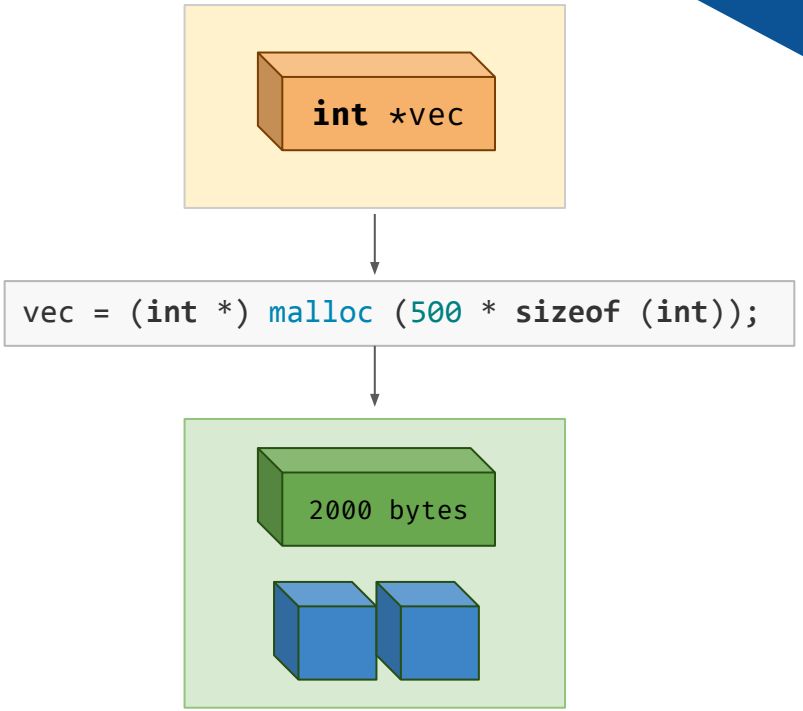
- Es necesario un puntero al tipo de dato del cual queremos pedir memoria.
- De no poder ser necesario devolverá un puntero con valor NULL.
- Al devolver un puntero void será necesario castear el valor devuelto al tipo de dato de nuestra necesidad.
- Luego de utilizar la memoria dinámica. Es necesario liberarla con la función free. Así, el sistema operativo podrá reutilizarla.

malloc

```
#include <cstdlib>

int main(){
  /* Memoria dinámica para
  un vector de 500 elementos */
  int *vec;
  vec = (int *) malloc (500 * sizeof (int));
  if (vec == NULL)
    exit(1); // No hay memoria

  // Resto del programa
  free (vec);
}
```



Ejemplos en C/C++