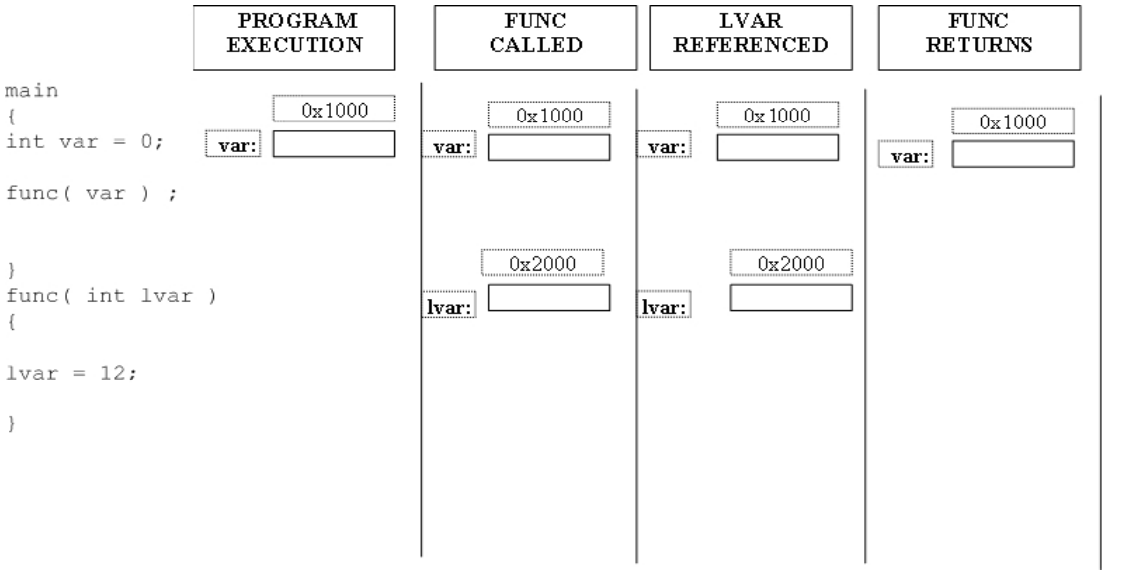
Working With Memory

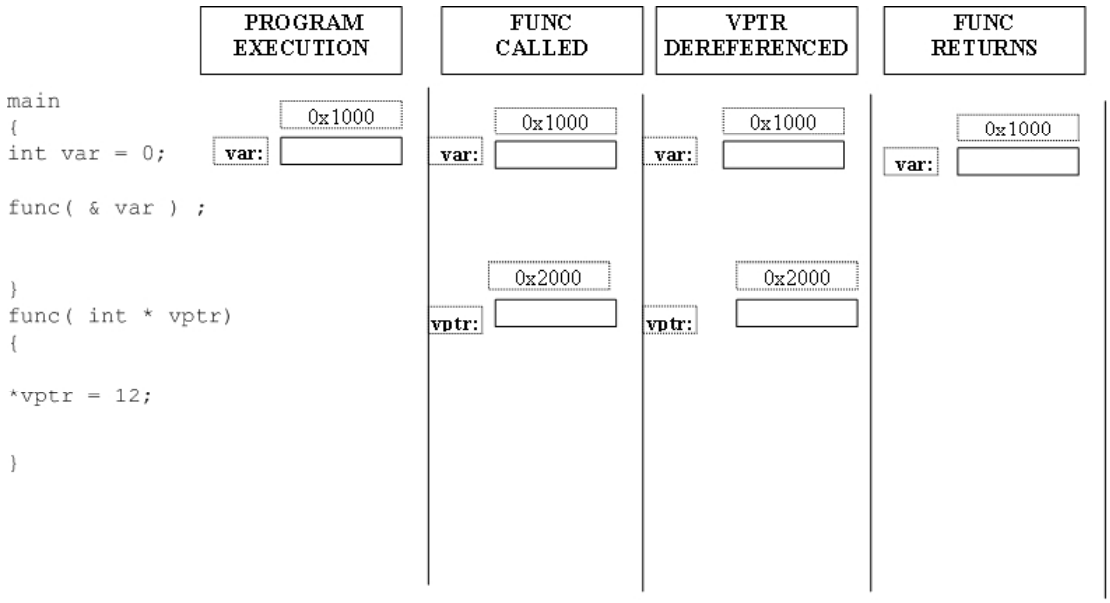
Working With Memory Topics

- Pass By Value
- Pass By Reference
- Dynamic Memory Allocation
- Passing Pointer To Pointer
- Related Memory Functions

Pass By Value



Pass By Reference



Dynamic Memory Allocation

- #include <stdlib.h>
- void *malloc(size_t size);
  - Allocates *size* bytes of memory
  - Returned pointer is "suitably aligned" for any data type
  - Returns *NULL* if no memory is available
- void *calloc(size_t count, size_t size);
  - Allocates *count* * *size* bytes of memory
  - Returns *NULL* if no memory is available
  - Memory is aligned the same way as *malloc*
  - Memory is initialized to zeros

Dynamic Memory Allocation

- void free(void *old_ptr);
  - frees previously allocated memory
- void *realloc(void *old_ptr, size_t size);
  - If *old_ptr* is *NULL*, behaves like *malloc*
  - If *old_ptr* is not *NULL*, and size is 0, behaves like *free*
  - Otherwise attempts to resize a previously allocated area
  - May return a pointer to a new area
  - Returns *NULL* if area can't be resized

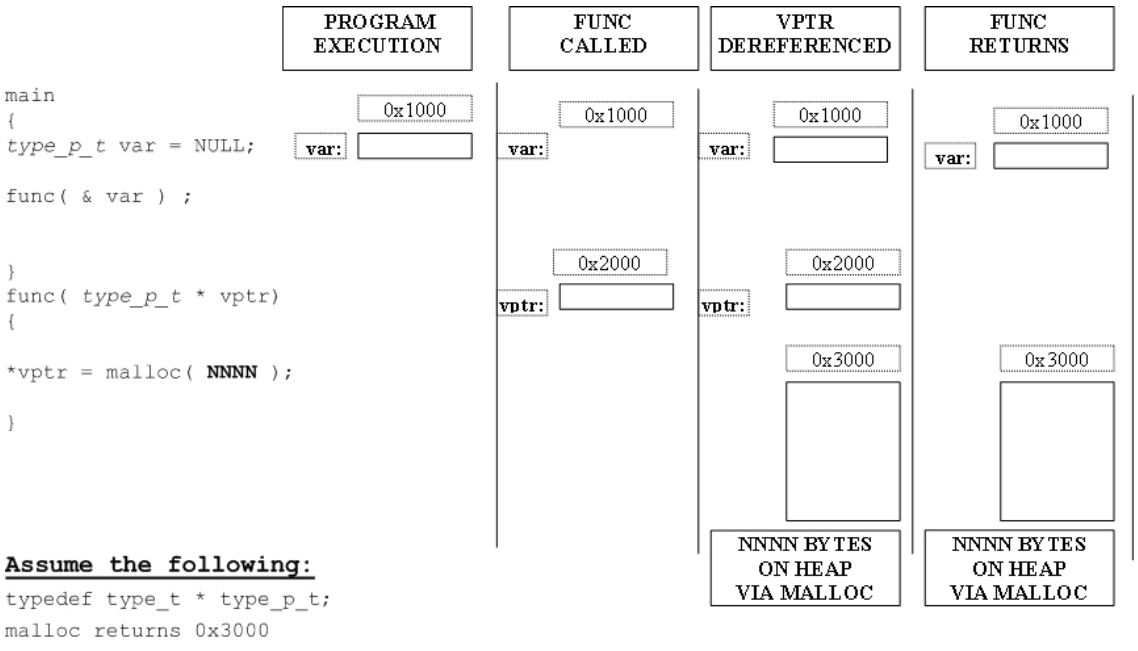
Dynamic Memory Allocation

- Common Errors
  - Forgetting to check for allocation failure
  - Requesting 0 bytes of memory
  - Reallocating a position-dependent data structure
  - Reading from uninitialized memory
  - Leaking memory

Dynamic Memory Allocation

- Common Errors
  - Freeing the same memory twice
  - Reading/writing beyond the end of allocated memory
  - Memory fragmentation
  - Assumptions about zero-initialization
  - Avoiding the pitfalls:

Passing Pointer To Pointer



Related Memory Functions

- **#include <string.h>**
- **int memcmp(const void * ptr1,**
  const void *ptr2, int len);
  - compares *len* bytes pointed to by *ptr1* with *len* bytes pointed to by *ptr2*
  - returns lexicographical difference
  - returns 0 if *len* bytes are identical

Related Memory Functions

- **void * memcpy(void * dest,**
  const void * src, int len);
  - copies *len* bytes from *src* to *dest*
  - may **not work correctly** if *dest* and *src* overlap
  - faster then memmove
- **void * memmove(void * dest,**
  const void * src, int len);
  - copies *len* bytes from *src* to *dest*
  - **works correctly** if *dest* and *src* overlap