red – statically sized, nested array
blue – dynamically sized, nested array
green – statically sized, multilevel array

```
#define SIZE 10
#include <stdlib.h>
#include <stdio.h>

long red[SIZE][SIZE];
long *blue;
long *green[SIZE];
int n;
```

```
long foo(long i, long j, long val) {

  long x,y;

  blue=(long *)calloc(n*n,sizeof(int));

  for (x=0; x<SIZE; x++)
    {
      green[x]=calloc(SIZE,sizeof(int));
      for (y=0; y<SIZE; y++)
        {
          red[x][y]=rand();
          green[x][y]=rand();
          blue[x*n+y]=rand();
          calloc(rand()%10,sizeof(int));
        }
    }

  setred(i,j,val);
  setgreen(i,j,val);
  setblue(i,j,val);

}
```

```
void setred(long i, long j, long val)
{
  red[i][j]=val;
}

void setblue(long i, long j, long val)
{
  blue[i*n+j]=val;
}

void setgreen(long i, long j, long val)
{
  green[i][j]=val;
}
```

```
0000000000400554 <setred>:
  400554:       48 8d 04 bf             lea    (%rdi,%rdi,4),%rax
  400558:       48 8d 04 46             lea    (%rsi,%rax,2),%rax
  40055c:       48 89 14 c5 60 0c 60    mov    %rdx,0x600c60(,%rax,8)
  400563:       00
  400564:       c3                      retq

0000000000400565 <setblue>:
  400565:       48 63 05 14 0a 20 00    movslq 0x200a14(%rip),%rax        # 600f80 <n>
  40056c:       48 0f af f8             imul   %rax,%rdi
  400570:       48 01 fe                add    %rdi,%rsi
  400573:       48 8b 05 0e 0a 20 00    mov    0x200a0e(%rip),%rax        # 600f88 <blue>
  40057a:       48 89 14 f0             mov    %rdx,(%rax,%rsi,8)
  40057e:       c3                      retq

000000000040057f <setgreen>:
  40057f:       48 8b 04 fd 00 0c 60    mov    0x600c00(,%rdi,8),%rax
  400586:       00
  400587:       48 89 14 f0             mov    %rdx,(%rax,%rsi,8)
  40058b:       c3                      retq
```

```
struct node_t {
  char x;
  long y;
  short z;
};

struct node_t *a[4];
```

Creating a 4x4 array of nodes

```
ptr=(struct node_t *)malloc(4*sizeof(struct node_t));

a[i]=ptr;
```

```
(gdb) print &a
$1 = (<data variable, no debug info> *) 0x600b40
(gdb) x/32xb 0x600b40
0x600b40 <a>:    0xe0    0x10    0x60    0x00    0x00    0x00    0x00    0x00
0x600b48 <a+8>: 0x10    0x12    0x60    0x00    0x00    0x00    0x00    0x00
0x600b50 <a+16>:        0x70    0x11    0x60    0x00    0x00    0x00    0x00    0x00
0x600b58 <a+24>:        0x30    0x10    0x60    0x00    0x00    0x00    0x00    0x00
```

```
struct node_t {
  char x;
  long y;
  short z;
};
```

```
(gdb) x/96xb 0x601030
0x601030:       0x41    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x601038:       0xef    0xbe    0xad    0xde    0xef    0xbe    0xad    0xde
0x601040:       0xed    0xfe    0x00    0x00    0x00    0x00    0x00    0x00
0x601048:       0x2f    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x601050:       0xef    0xbe    0xad    0xde    0xef    0xbe    0xad    0xde
0x601058:       0xed    0xfe    0x00    0x00    0x00    0x00    0x00    0x00
0x601060:       0x44    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x601068:       0xef    0xbe    0xad    0xde    0xef    0xbe    0xad    0xde
0x601070:       0xed    0xfe    0x00    0x00    0x00    0x00    0x00    0x00
0x601078:       0x65    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x601080:       0xef    0xbe    0xad    0xde    0xef    0xbe    0xad    0xde
0x601088:       0xed    0xfe    0x00    0x00    0x00    0x00    0x00    0x00
```

```
struct node_t {
  char x;
  long y;
  short z;
};
```

```
struct node_t {
  long y;
  char x;
  short z;
};
```

```
(gdb) print &a
$1 = (<data variable, no debug info> *) 0x600b40
(gdb) x/32xb 0x600b40
0x600b40 <a>:    0xc0    0x10    0x60    0x00    0x00    0x00    0x00    0x00
0x600b48 <a+8>:  0xb0    0x11    0x60    0x00    0x00    0x00    0x00    0x00
0x600b50 <a+16>:         0x30    0x11    0x60    0x00    0x00    0x00    0x00    0x00
0x600b58 <a+24>:         0x30    0x10    0x60    0x00    0x00    0x00    0x00    0x00
(gdb) x/96xb 0x601030
0x601030:        0xef    0xbe    0xad    0xde    0xef    0xbe    0xad    0xde
0x601038:        0x41    0x00    0xed    0xfe    0x00    0x00    0x00    0x00
0x601040:        0xef    0xbe    0xad    0xde    0xef    0xbe    0xad    0xde
0x601048:        0x2f    0x00    0xed    0xfe    0x00    0x00    0x00    0x00
0x601050:        0xef    0xbe    0xad    0xde    0xef    0xbe    0xad    0xde
0x601058:        0x44    0x00    0xed    0xfe    0x00    0x00    0x00    0x00
0x601060:        0xef    0xbe    0xad    0xde    0xef    0xbe    0xad    0xde
0x601068:        0x65    0x00    0xed    0xfe    0x00    0x00    0x00    0x00
0x601070:        0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x601078:        0x41    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x601080:        0x69    0x98    0x3c    0x64    0x73    0x48    0x33    0x66
0x601088:        0x51    0xdc    0xb0    0x74    0xff    0x5c    0x49    0x19
```