

# The Tromino Tiling Puzzle (I)

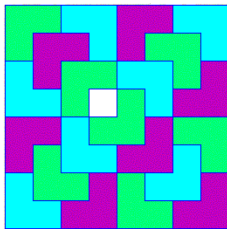
— Pointers, (2D-)Arrays and Recursion

魏恒峰

hfwei@nju.edu.cn

2017 年 11 月 03 日

```
int (*pa)[n] = malloc( sizeof(int[m][n]) );
```



# Memory Model

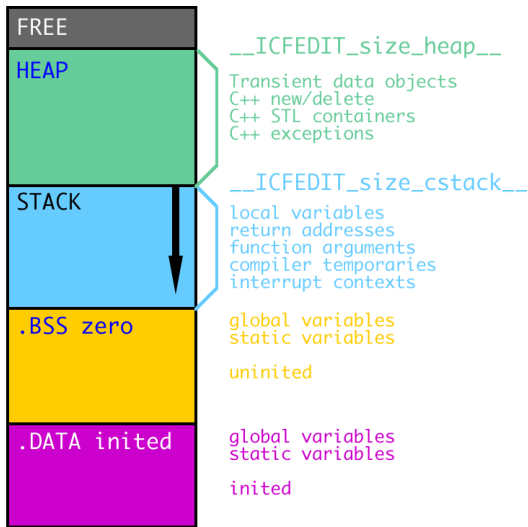
## Definition (Memory (K&R))

The memory is organized as a collection of consecutively **addressed** cells that may be manipulated **individually** or in **contiguous groups**.

**address of  
memory cell**    *RAM (memory)*

000...000	00001101
000...001	00000011
000...010	00000000
000...011	00101101





Program Memory

Type	Scope	Lifetime	Storage
<i>Global</i>	The entire file	The lifetime of the <b>program</b>	.BSS/.DATA
<i>Static</i>	The <b>function</b> it is declared within	The lifetime of the <b>program</b>	.BSS/.DATA
<i>Automatic</i>	The <b>function</b> it is declared within	While the <b>function</b> is executing	Stack
<i>Dynamic</i>	Determined by the <b>pointers</b> that reference this memory	Until the memory is <b>freed</b>	Heap

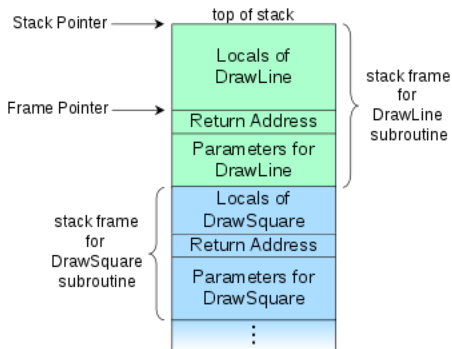
```

int a = 1;    // global (inited)
int b;       // global (uninited)

int f(void) {
    int c = 0;           // automatic (local)
    static int d = 0;   // static (inited)
    d++;

    int *p = malloc( sizeof(int) ); // dynamic
    free(p);
}

```



```
void DrawSquare(int len) {
    ...
    DrawLine(len, dir);
    ...
}
```

# Pointers and Arrays

*In C, there is a **strong relationship between pointers and arrays**, strong enough that pointers and arrays should be discussed simultaneously.*

— K&R



# Pointers

## Definition (Pointers (K&R))

A pointer is a **variable** that contains the **address** of a variable.

```
int a = 0;  
int *p = &a;
```

## Definition (Pointers in Memory (K&R))

A pointer is a group of cells (often two or four) that can hold an address.

```
swap(a, b);

void swap(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}
```

*Pointer arguments enable a function to access and change objects in the function that called it.* — K&R

```
swap(&a, &b);

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

# 1D Arrays

## Definition (Name of an Array)

The **value** of a **variable of type array** is the **address** of **element zero** of the array.

$$a \triangleq \&a[0]$$

array-1d.c

```
int a[5];  
a,    &a[0] // what are they?  
  
int *pa = a;  
int *pa = &a[0];  
  
&a // what is this?
```

```
int a[5];
```

```
int *pa = a;
```

Definition (Equivalence between Accesses)

$$pa[i] \triangleq a[i] \triangleq *(a + i)$$

*When an array name is passed to a function, what is passed is a **pointer**, the location of the initial element.*

— K&R

```
void f(int a[5])
void f(int a[], int n);
void f(int *a, int n);

f(a, 5); // int a[5] = {0};

f(pa, 5); // int *pa = a;
```

```
int a[n];
f(a, n);

int *pa = malloc( sizeof(int[n]) );
f(pa, n);
```

# 2D Arrays



```
int a[3][5] = {  
    {1,2,3,4,5},  
    {6,7,8,9,10},  
    {11,12,13}  
};
```

*Elements (of an 2D array) are stored by rows.*

— K&R

array-2d.c (Part I)

*In C, a 2D array is really a 1D array, each of whose elements is an array.*

— K&R

```
a,    &a[0],    a[0],    &a[0][0],    &a
int (*pa)[5] = a; // a pointer to an array of 5
integers
```

array-2d.c (Part II)

```
a[i][j] // *((*(a + i)) + j)
```

```
void f(int a[3][5]);  
void f(int a[][5], int m); // m rows  
void f(int (*a)[5], int m);  
  
f(a, 3); // int a[3][5];  
f(pa, 3); // int (*pa)[5] = a;
```

```
void f(int m, int n, int a[m][n]);  
void f(int m, int n, int a[][n]);  
void f(int m, int n, int (*a)[n]);
```

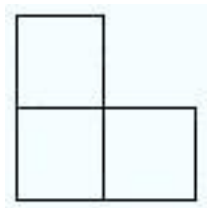
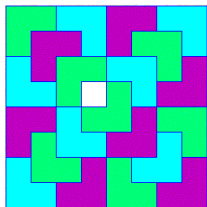
```
int a[m][n];  
f(m, n, a);
```

```
int (*pa)[n] = malloc( sizeof(int[m][n]) );  
f(m, n, pa);
```

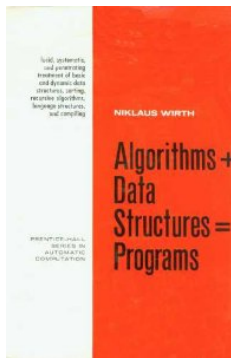
# The Tromino Tiling Puzzle

## Theorem (Tromino Tiling Theorem)

*For any positive integer  $k$ , a  $2^k \times 2^k$  checkerboard with any one square removed can be tiled using right trominoes.*



 Play with the Interactive Tromino Puzzle



tromino-tiling-vla.c

Thank  
You!