The Tromino Tiling Puzzle (I)

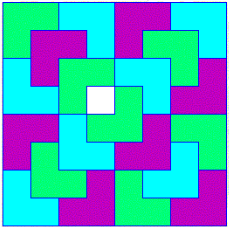
— Pointers, (2D-)Arrays and Recursion

魏恒峰

hfwei@nju.edu.cn

2017 年 11 月 03 日

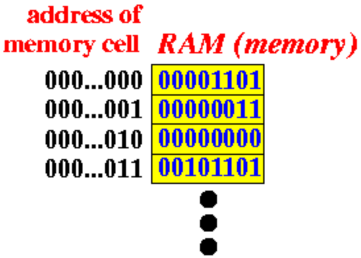
```c
int (*pa)[n] = malloc( sizeof(int[m][n]) );
int (*pa)[n] = malloc( sizeof(int[m][n]) );
```

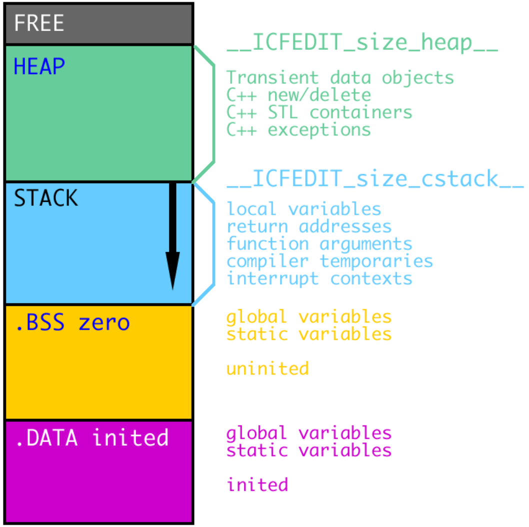


Memory Model

Definition (Memory (K&R))

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

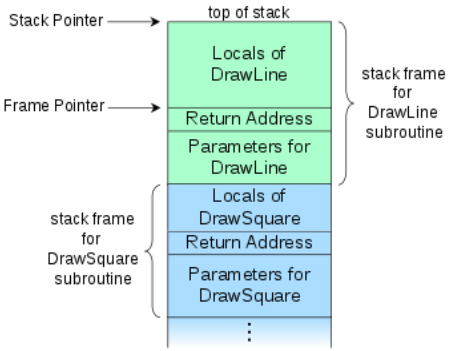




Program Memory

Type	Scope	Lifttime	Storage
Global	The entire file	The lifetime of the program	.BSS/.DATA
Static	The function it is declared within	The lifetime of the program	.BSS/.DATA
Automatic	The function it is declared within	While the function is executing	Stack
Dynamic	Determined by the pointers that reference this memory	Until the memory is freed	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 (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 = tmp;
}
```

```c
swap(a, b);

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

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 = tmp;
}
```

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 = tmp;
}
```

1D Arrays

Definition (Name of an Array)

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

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

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?
```

```c
    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

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

```c
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*

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)

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)
```

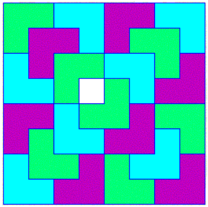
```
void f(int a[3][5]);
void f(int a[][5], int m); // m rows
void f(int (*a)[5], int m);

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

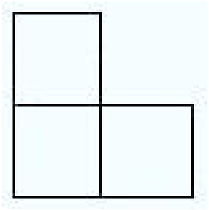
```
void f(int m, int n, int a[m][n]);
void f(int m, int n, int a[][n]);
void f(int m, int n, int (*a)[n]);

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

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

The Tromino Tiling Puzzle

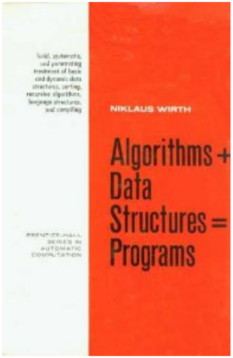
Theorem (Tromino Tiling Theorem)

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


lucid, systematic, and penetrating treatment of basic and dynamic data structures, sorting, recursive algorithms, language structures, and compiling
NIKLAUS WIRTH
Algorithms + Data Structures = Programs
PRENTICE-HALL SERIES IN AUTOMATIC COMPUTATION

`tromino-tiling-vla.c`

Thank
You!