Multidimensional Arrays

- How do you make it into a continuous block of memory?
  
  • by rows or by columns

  ![Diagram showing multidimensional arrays]

  - row-major order
    
    ![Row-major order diagram]

  - column-major order
    
    ![Column-major order diagram]

- Again, we need random access in constant time. How do we compute the address of \(A[i][j]\)?
  
  • In row major order, it is
    
    \[ \text{addr of } A[i][j] = \text{addr-of-}A + i \times \text{slot-size} \times \text{row-size} + j \times \text{slot-size} \]

- Does row-major vs col-major make any difference? If you are searching the array which order should you do it? Why?
  
  • same as hardware; cache hits increase

- Which way does Java do it?
  
  • Neither. Advantage of Java way is you can have irregular shaped arrays.

  • Java has arrays of arrays. While a given array's entries are stored in a contiguous block memory, the subordinate arrays those entries point to are object references to complete separate unrelated blocks of memory.

  ![Java array of arrays diagram]

  So a \(a[2][3]\) means “Get the array referenced by the entry at index 2 of \(a\), then get the entry referenced by index 3 of that subordinate array.”
Type of Data in Arrays

- In statically typed languages, need all entries to be the same type. Why?
  - To avoid having to type cast and so that all slots have same size so can calculate the index.
- Then how do dynamically typed languages handle arrays? Since you can put any size of data in each slot, how can the interpreter or compiler calculate $A[i]$ efficiently?
  - Each slot contains pointers to the data and all pointers are the same size.

Functions as data types

- Functions can also be a data type that can be passed as parameters and assigned to variables
- Why would you want to pass a function as parameters?
  - convenience; polymorphism in C
- Example - Python (show functions.py, ask the outputs)

```python
def foo(x):
    print 3+x

bar = foo
bar(5) #prints 8

def baz(fn):
    fn(6)

baz(bar) #prints 9
```

- Easy with Python, but for statically typed languages, you have to declare the type of the function. How do you specify the function type?
  - Example - C (show functionPointer.c)

```c
#include <stdio.h>

void print_int (int i) {
    printf("%d\n", i);
}

int main() {
    // declare a function pointer
    void (*func)(int);

    // initialize a function pointer
    func = print_int;

    // call a function pointer
    func(10);

    return 0;
}
```
• Example - C qsort (show qsortExample.c)
  - #include <stdlib>
  - void qsort(void *base, size_t nmemb, size_t size, int(*compar)
    (const void *, const void *));

/**
 * Example of qsort: sort parts of a drone in ascending order in terms of price
 * Ying Li
 * 09/21/2020
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
    char name[50];
    int price;
} part;

int compare (const void* p1, const void* p2) {
    int m = ((part *)p1)->price;
    int n = ((part *)p2)->price;
    return m-n;
}

int main () {
    part drone[6];

    strcpy(drone[0].name, "DJI Flamewheel Kit");
    drone[0].price = 299;

    strcpy(drone[1].name, "Navio2");
    drone[1].price = 117;

    strcpy(drone[2].name, "Power Module");
    drone[2].price = 22;

    strcpy(drone[3].name, "Receiver");
    drone[3].price = 30;

    strcpy(drone[4].name, "Batteries");
    drone[4].price = 60;

    strcpy(drone[5].name, "Sonar");
    drone[5].price = 25;

    qsort (drone, 6, sizeof(part), compare);

    for (int i = 0; i < 6; i++) {
        printf("%s: %d\n\n", drone[i].name, drone[i].price);
    }

    return 0;
}
• return value of compar function
  < 0 the first parameter is smaller than the second one (first placed before second)
  = 0 two parameters are the same
  > 0 the first parameter is larger than the second one (second placed before first)