APS 105 Lecture 17 Notes

Last lecture: Arrays and Pointers

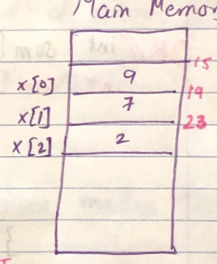
Today: Pointer Arithmetic and Dynamic memory Allocation

Recap: Key notion is that the array identifier acts as a pointer to the 1st element in the array

```
int x[] = {9, 7, 2};
```

x ⇔ &x[0]

*x ⇔ x[0]



How about x[1], do I have a pointer to it?

Yes

Pointer Arithmetic

(x + 1) ⇔ &x[1]

*(x + 1) ⇔ x[1]

Imp:

```
if int *q = &x[2];
int dist = q - x;
//dist = (23 - 15) / sizeof(int) = 2
```

x is the address of the 1st element of the array, so adding 1 adds to the address 1, so I move to the next block, which is the 2nd element of my array

- DEMO - "refer to source code posted"

```
OR // int sumData (int [], int);
   int sumData (int *, int);
   int main (void) {
        int data [3] = {9, 7, 2};
        int sum = sumData (data, 3);
        return 0;
   }

OR // int sumData (int inputData [], int size) {
   int sumData (int *inputData, int size) {
        int sum = 0;
        for (int i = 0; i < size; i++) {
             sum += inputData [i];
             // OR sum += *(inputData + i);
        }
```

OR

Dynamic Memory Allocation:

If I have an array that holds marks `double marks[10];` (will hold 10 marks)

What if I don't know # of students / marks before writing the code?

Way I: `double marks [1000];` (max # of marks in any class)

Problem: this is a waste of memory as 1000 double variables will be reserved

Way II : dynamic memory allocation
↳ request a chunk of memory of specific size

* malloc : a special C function from stdlib to acquire any size of memory
(memory allocation)

* free : a function in stdlib to return memory when no longer needed, so you don't run out of memory

Hence, to use malloc and free, must

```
#include <stdlib.h>
```

∴ What is the unit of memory does malloc take?

BYTES

Recall: int typically takes 4 Bytes
double " " 8 Bytes
char 1 Byte
bool 1 Byte

OR you can use sizeof(<type>) to know # of bytes to store <type>

E.g.

```
int numOfStudents;
int *marksArray;
// This will be the pointer to my 1st element of my dynamically allocated array
printf("Enter # of marks");
scanf("%d", &numOfStudents);
marksArray = (int*) malloc (numOfStudents * sizeof(int));
```

entered by user

4

# of bytes to reserve

(int*) returns pointer to the 1st byte of the allocated memory.

need to type cast to (int*)

type is (void*)

```
printf("Enter marks: ");
for (int i=0; i<numOfStudents; i++)
    scanf("%d", marksArray+i);
!
!
free (marksArray);   // returns the amount of memory for reuse
marksArray = NULL;   // marksArray: DON'T USE
```