2510 : Lecture 3 Arrays II

January 13, 2017

Cont. Processing Arrays

③ Testing whether an array on integers consists entirely of non-negative integers

- there is no BOOLEAN type in C

equivalence
0.0
ANSIIC code 0

→ Ø false, everything else is true

```
ex. if ("hello world") {   // true
      ...
    }
```

- it's common to have a function return an int if it needs to return a boolean value

should we test for positive or negative numbers?
ans: negative

```
                                  ┌ because we aren't changing
int arr.all.nonneg(const int a[], size_t n){
    size_t i;
    for (i = Ø; i<n; i++)
      if (a[i] < Ø)   /* testing for negatives */
          return Ø;   /* "return early */
    return 1;
```

2510 : Lecture 3 Arrays ||

④ Looking for an integer in an integer array

- return the position in array

```
size_t   arr_find (const int a[], size_t n, int x) {
             size_t i;
             for (i = 0; i < n; i++)
                 if (a[i] == x)
                     return i;
                 return -1;      /* alternative : return n; */
             }
```

if no int is found, we need to return failure number → return -1;

- size_t is an unsigned type ; under 2's complement
  -1 = 1111

  1. As an unsigned, it becomes the biggest positive numbers
     ∴ -1 is an impossible index

⑤ Double the numbers in an int array

- doesn't return a number, just change value

```
void arr_double (int a[], size_t n) {
    size_t i;
    for (i = 0; i < n; i++)
        a[i] *= 2;
}
```

2510 : Lecture 3 Arrays + Tests

- **Testing**
  - ↳ using example arr_sum for a specific array; this is not a good way to test :

    Printing the return values isn't telling us if they are correct!

  - ↳ INSTEAD : print whether function gives the correct answer

```
int a [] = {3, 2, 7, 6, 8};
printf ("%d\n", arr_sum (a, 5) == 26);
    /* prints 1 if arr_sum gives the correct
       value */
```

~~but what if we have multiple tests, how can we tell which one failed?~~

[still may be difficult to figure out which test failed.]

▷ Macro - we'll use a macro to print the test after the usual stuff at the beginning

all on one line →
```
#define CHECK (PRED) printf ("%s... %s\n",
    (PRED) ? "passed" : "FAILED", #PRED)
```
how do we actually use this?

```
int a [] = {3, 2, 7, 6, 8};
CHECK (arr_sum (a, 5) == 26);
```

possible output :

```
passed... arr_sum (a, 5) == 26
FAILED... arr_sum (a, 5) == 26
```

```
CHECK (arr_find (a, 5, 1) == (size_t) - 1);
```
cast it

2510 : Lecture 3

- **Pre Conditions**
  - a pre-condition on a function is the condition that must hold before the function will work correctly. (number of elements must be positive)

ex. maximum of an array

```
/* precondition : n > ∅ */
if arr-~~sum~~max(const int a [], size_t n){
    size_t i;
    int         max = a[∅]; /* possible b/c n > ∅ */
    for (i = ∅; i<n; i++)
        if (a[i] > max)
            max = a[i];
    return max;
}
```

if a[i] big than max...a[i] is Max

What does it mean to work correctly?