2510 : Lecture 2 Arrays II

January 11, 2017

Arrays

Examples:

used most of time

1) `int a[10];` /* array of 10 ints contains random values if local */

2) `int b[3] = {3, 2, 7};`
initializers

3) `int c[5] = {3, 2, 7};` /* rest default to 0 */

4) `int d[3] = {3, 2, 7, 2, 8};` /* error - too many */

5) `int e[] = {3, 2, 7};` /* compiler will count */

Standard idiom to Process Array

```
T a[N]; /* array */
```

T: some type; N: positive integer

```
size_t i;
for (i = 0; i < N; i++)
    /* process a[i] */
```

↳ in ANSI C, variable must be declared at the beginning of a block

eg.
```
int main(void) {
    int x = 1;
    printf("%d\n", x);
    int y = 2; /* error: not the start of a block */
```

2510 : Lecture 2 Arrays

cont.

```
if (x > 0) {
    int z = 3;  /* ok, because new block */
    ...
}
int z = 4;  /* error: not at start of block */
...
}
```

What is size_t ? (t = type)

↳ It's the name of a type. TYPE to store sizes

↳ It's some unsigned integer type

↑ only non-negative

eg. Using Standard Idiom

(1) summing integer in an integer array

```
int main(void) {
    int a[] = {3, 2, 7, 6, 8};
    size_t i;
    int sum = 0;
    for (i = 0; i < 5; i++)   /* cannot do arr.length
      sum += a[i];                b/c that is OOP */
    printf("%d\n", sum);
    return 0;
}
```

2510 : Lecture 2 Arrays II

Examples : Using Standard Idiom

② Put the summary in a function

↳ in C, everything is passed by value (not reference)
eg. the function gets a copy of the argument you pass into it.

↳ impossible to pass the whole array by itself into a function

reference to a by ...

```
int a[10];
a ? ← in most cases; this used as starting address of array
a[10] ? /* 11th element; out of bounds */
```

↳ functions that process arrays typically need at least two parameter:
① starting address of the array
② number of elements in the array

eg. Array summing

```
int arr_sum (const int a[], size_t n) {
  size_t i;
  int sum = 0;
  for (i = 0; i < n; i++)
    sum += a[i];
  return sum;
}
int main(void) {
  int a[] = {3, 2, 7, 6, 8};
  printf("%d\n", arr_sum(a, 5));
  return 0;
}
```

const: prevent from change array element

n: # of elements

standard loop