Arrays (Contd.)

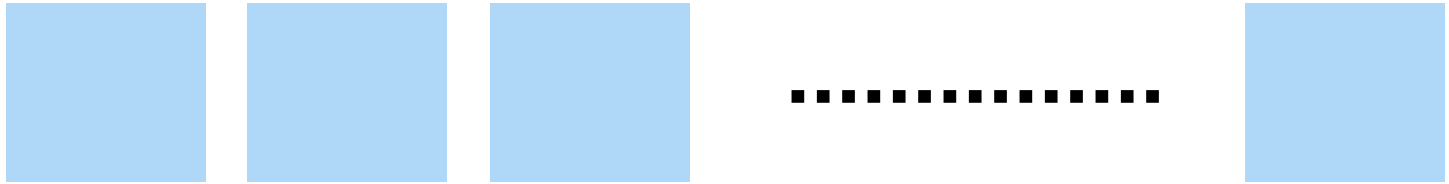
ESC101: Fundamentals of Computing

Nisheeth

Recap: Arrays

- A collection of elements all of which have the same data type

```
float marks[500];
```

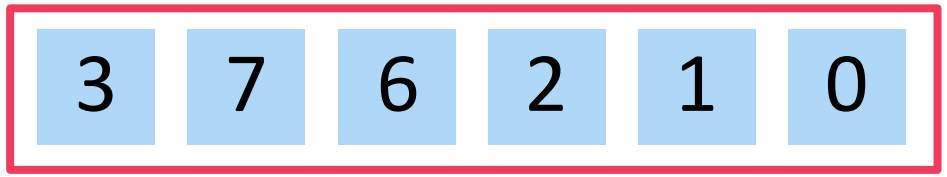


- Each array element is accessed using the array index (integer-valued)
  - For the above example, marks[0], marks[2], marks[499], marks[int_expr] where int_expr is integer-valued expression such that 0 <= int_expr <= 499

Recap: Array: Declaration and Initialization

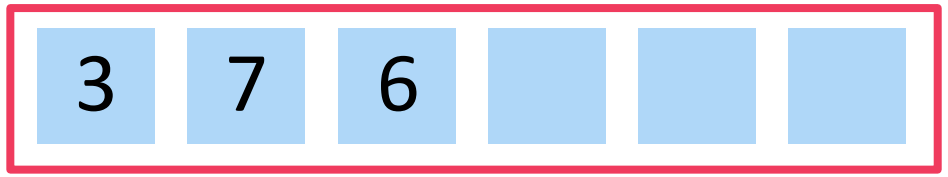
Can be initialized at time of declaration itself

```
int a[6] = {3,7,6,2,1,0};
```



Can be partly initialized as well

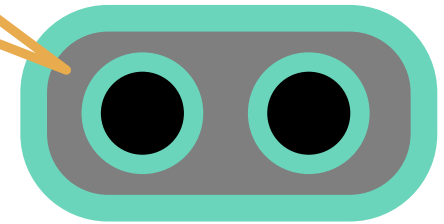
```
int a[6] = {3,7,6};
```



Over initialization may crash

```
int a[6] = {1,2,3,4,5,6,7,8,9};
```

I will figure out how much space needed



No need to specify the array size during declaration

Better way is the following

```
int a[] = {1,2,3,4,5,6,7,8,9};
```

Warning: uninitialized arrays contain garbage, not zeros

Array: Declaration and Initialization

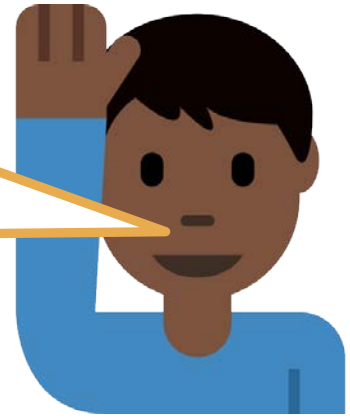
- Can declare the array first and initialize its elements later
- The later initialization can be done using user-provided values (e.g., using scanf), or some expression, or using some fixed values

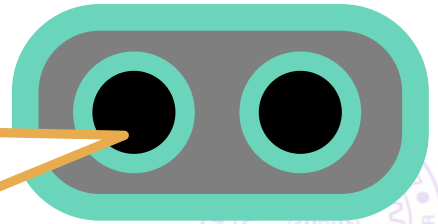
```
int i,tmp,a[5];
for(i=0;i<5;i++){
    scanf("%d",&tmp);
    a[i] = tmp;
}
```

Read a use-provided value

Assign the read value to the i^{th} element of the array

Can I run the loop as for(i=1;i<=5;i++)?

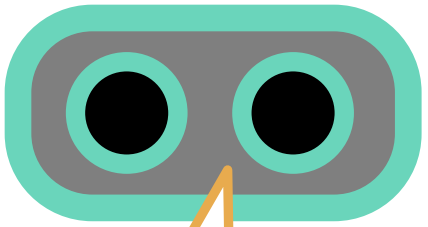




Yes, if you use a[i-1] = tmp; in loop's body

Array: Declaration and Initialization

- Can declare the array first and initialize its elements later
- The later initialization can be done using user-provided values (e.g., using scanf), or some expression, or using some fixed value



A shortcut for reading user provided values

```
int i,a[5];
for(i=0;i<5;i++){
    scanf("%d",&a[i]);
}
```

Note: &a[i] is evaluated as &(a[i]) since [] has higher precedence than &

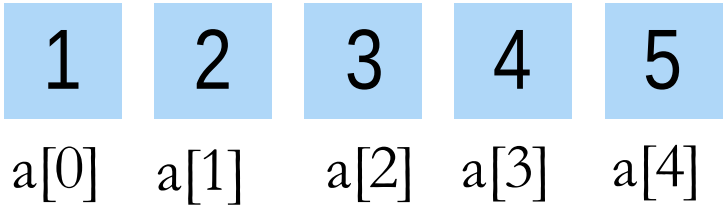
Directly read a user provided value into the i^{th} element of the array (the tmp variable is not needed)

Operator Name	Symbol/Sign	Associativity
Brackets, **array subscript**, Post increment/decrement	(), [] ++, --	Left
Unary negation, Pre increment/decrement, NOT	-, ++, --, !	Right
Multiplication/division/ remainder	*, /, %	Left
Addition/subtraction	+, -	Left
Relational	<, <=, >, >=	Left
Relational	==, !=	Left
AND	&&	Left
OR	||	Left
Conditional	? :	Right
Assignment, Compound assignment	=, +=, -=, *=, /=, %=	Right

Array: Declaration and Initialization

- Can declare the array first and initialize its elements later
- The later initialization can be done using user-provided values (e.g., using scanf), or some expression, or using some fixed value

```
int i,a[5];
for(i=0;i<5;i++){
   a[i] = i+1;
}
```

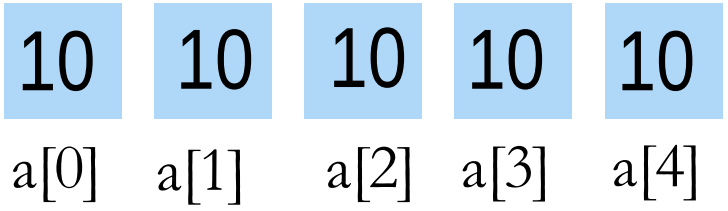


Assign a value of expression i+1 to the i^{th} element of the array

Array: Declaration and Initialization

- Can declare the array first and initialize its elements later
- The later initialization can be done using user-provided values (e.g., using scanf), or some expression, or using some fixed value

```
int i,a[5];
for(i=0;i<5;i++){
   a[i] = 10;
}
```



Assign a fixed (constant) value 10 to the i^{th} element of the array

Tracing the execution of an array based program

```
include <stdio.h>
int main () {
      int a[5];
      int i;
       for (i=0; i < 5; i= i+1) {
            a[i] = i+1;
        }
        return 0;
}
```

Let us trace the execution of the program.

i 5

a[0]	a[1]	a[2]	a[3]	a[4]
1	2	3	4	5

Statement becomes a[0] =0+1;
Statement becomes a[1] =1+1;
Statement becomes a[2] =2+1;
Statement becomes a[3] = 3+1;
Statement becomes a[4] = 4+1;

Arrays: Some Example Programs

- Create an integer array of size 100
- Initialize elements with even index as 0
- Initialize elements with odd index as 1

Method 1

```
int i,a[100];
for(i=0; i<100; i=i+1){
   if(i%2==0) a[i] = 0;
   else a[i] = 1;
}
```

Arrays: Some Example Programs

- Create an integer array of size 100
- Initialize elements with even index as 0
- Initialize elements with odd index as 1

```
int i,a[100];
for(i=0; i<100; i=i+2){
   a[i] = 0;
   a[i+1] = 1;
}
```

Incrementing the loop counter by 2

Method 2, without if-else

This for loop will run 50 times. Each iteration will assign values to 2 elements, one at odd index, one at even index

Arrays: Some Example Programs

- Check whether a sequence of numbers is a palindrome sequence

Palindrome: Forward and Reverse gives the same sequence

Some palindromes:
1 2 3 4 5 4 3 2 1
1 2 3 3 2 1

Some non-palindromes:
1 2 3 4 5
1 2 3 3 4 1
9 0 4 0 8

Greek origin word:
palin = again,
dromos = direction

```
int main(){
   int a[100], temp, len = 0, i, flag = 1;

   while(1){
      scanf("%d", &temp);
      if(temp == -1)
         break;
      a[len++] = temp;
   }

   for(i = 0; i < len; i++)
      if(a[i] != a[len-i-1])
         flag = 0;
   if(flag) printf("YES");
   else printf("NO");

   return 0;
}
```

Let's specify a maximum sequence size

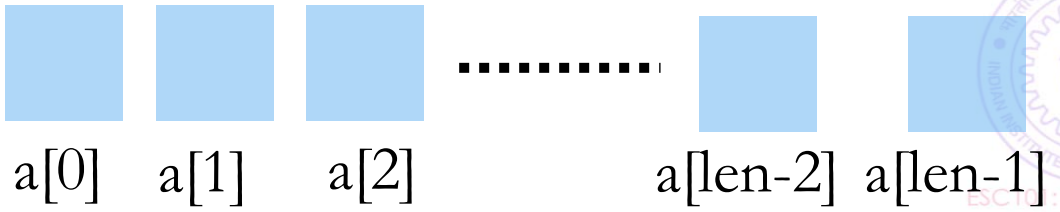
flag = 1 assumes that sequence is palindrome (set 0 if later found otherwise)

The while(1) loop keeps reading numbers until user enters -1, store each number as an element of the array named a

This line does a[len] = temp; and then increments len

After the while(1) loop exits, len is the size of the array (indices are 0 to len-1)

Compares a[0] with a[len-1], then a[1] with a[len-2], and so on. If any pair does not match, set flag variable to 0



Arrays: Some Example Programs

Read until user has entered 100 chars or the end-of-file (EOF) special character has been read.

Now print the characters in reverse order

```
#include <stdio.h>
int main() {
      char s[100];
      int count = 0;
      int ch;
      int i;

      ch = getchar();
      while ( ch != EOF && count < 100) {
            s[count] = ch;
            count = count + 1;
            ch = getchar();
      }

      i = count-1;
      while (i >=0) {
            putchar(s[i]);
            i=i-1;
      }
      return 0;
}
```

```
/* the array of 100 char */
/* counts number of input chars read */
/* current character read  */
/* index for printing array backwards */
```

```
/*read_into_array */
```

```
/*print_in_reverse */
```

getchar() returns a single character entered by the user

putchar() prints a single character

Next Class

- Functions and arrays
- Passing by value
- Passing by reference