GNG1106
Fundamentals of Engineering Computation

Instructor: **Hitham Jleed**

University of Ottawa

Fall 2023 ~

Outline

1 Array

2 Search

- An array is a collection of variables of the same type, declared via
  `Type arrayName[arrayLength];`
  where `arrayLength` is the number of variables the array contains.
- Examples:
  `int a[5];`
  `double x[200];`
- In standard C, the length (or size) of the array must be known at the compiling time. That is, it can not be a variable!
  - "`int a[N];`" (where `N` is an int-typed variable) is NOT allowed.
- Once an array is declared, its length can NOT be changed.

Highlight

In modern C (C++), some of these restrictions are actually relaxed. But in this course, using array in any way violating the above rules is considered WRONG, even if the code compiles and runs correctly.

- The declaration of an array allocates a block of memory space (i.e., a block of contiguous bytes) for the variables in the array.
  - By "`int a[5];`" an array a containing 5 int-typed variables is declared, and a block of 4x5=20 bytes is reserved for the array `a`, (assuming that each int-typed value takes 4 bytes to store)
  - The 5 variables (or array elements) can be accessed as `a[0]`, `a[1]`, `a[2]`, `a[3]`, `a[4]`. Note that the first element is indexed by 0 not 1.
- Each element of an array, say `a[3]`, can be used in exactly the same way as a regular variable. For example:
  - `a[3]=5;`
  - `printf("%d", a[3]);`
  - `a[3]=a[3]+1;`
- To access an element of an array, the index of the array element can be any expression that evaluates to the desired integer. For example, `a[5-2]` is the same as `a[3]`.

Array Declaration with Initialization

There are two ways:

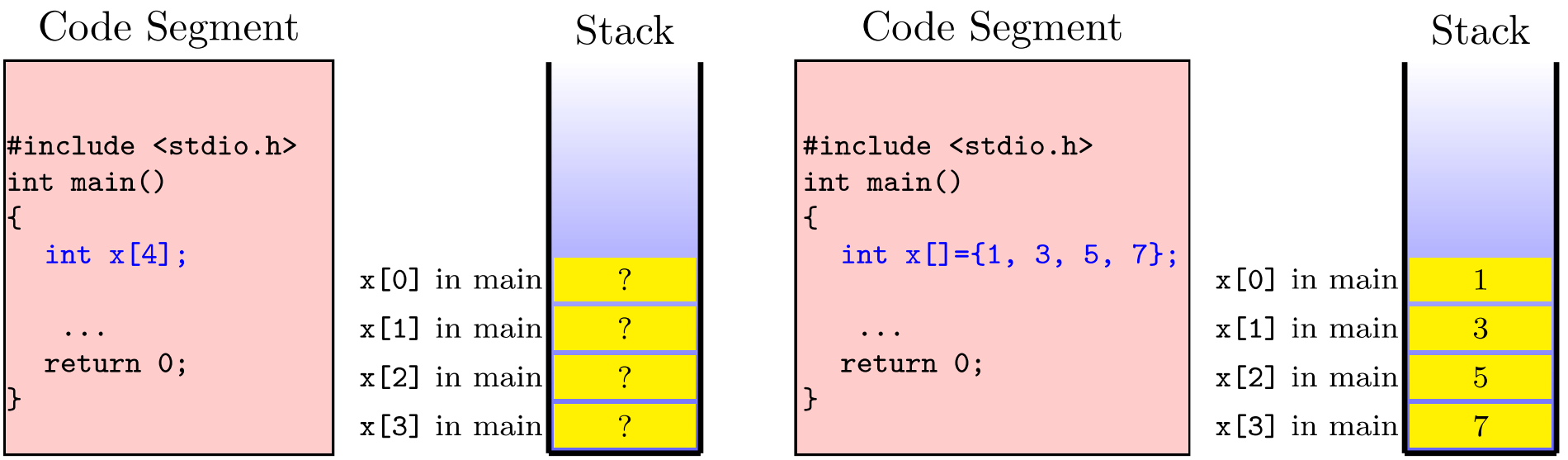
- `float x[5]={0.0, 0.5, 1.0, 1.5, 2.0};`
- `float x[]={0.0, 0.5, 1.0, 1.5, 2.0};`

Both ways will create an array of length 5 with name `x`; each element of the array is meant to store a float value, and the five elements are assigned values 0.0, 0.5, 1.0, 1.5, 2.0 respectively (in this order).

Note

If we declare and initialize an array by
`float x[5]={0.1, 0.2, 0.3};`
a float-typed array with name `x` and having length 5 will be created, and its first three elements will be assigned values 0.1, 0.2, 0.3 (in this order), the remaining two elements will be assigned value 0.
But we discourage the use of such a statement.

- The memory allocated for an array (declared inside a function) is in the stack segment of the program memory. That is, when an array is declared, a block of bytes is reserved in the stack to store the variables in the array.

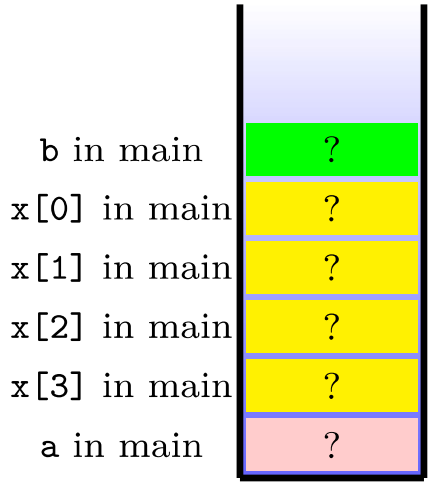


Code Segment

```
#include <stdio.h>
int main()
{
   int a;
   int x[4];
   int b;

    ...
   return 0;
}
```

Stack



Note

The stack picture in this example may not truthfully display the layout of variables `a`, `b` and array `x`. In reality, the memories for variables and arrays may not be allocated according to the order of their declaration. In addition, a "gap" of bytes may be skipped (i.e., not allocated). But any two consecutive elements of an array must be allocated two consecutive blocks of memory (without having a gap left in between).

The good news is: for most programming needs that most of you will ever encounter, this picture is sufficiently accurate.

The Name of an Array

Highlight

The name of an array is also the address of the first variable in the array! This is a very important fact!

```
#include <stdio.h>
int main()
{
    int x[3];
    printf("x= %p\n", x);
    printf("&x[0] = %p\n", &x[0]);
    return 0;
}
```

- The value of `x` is precisely the address of `x[0]`.
- This fact has an important consequence when an array is used as an input of a function.

Write a Program

Write a program that asks the user to enter 10 integers. After the user enters all 10 integers, the program prints the 10 numbers in the reverse order.

Coding Demonstration

Outline

1 Array

2 Search

Search

- Search is one of the basic operations often required in a computer program.
- We will use search in an integer array as an example to introduce two search algorithms.
- Problem statement: Given an array of n integers and an integer value, which is referred to as the key, if the key is contained in the array, find the index of the key in the array; otherwise declare that the array does not contain the key.
- I/O description:
  - Input: an array `A` of `n` integers and the key value `K`
  - Output: the index of `K` in `A` or `-1` if `K` is not contained in `A`

Linear Search Algorithm

- If the array `A` is not sorted, then one has to scan every element in the array and determine where the key is or whether it is in the array at all.
- This is known as linear search.
- The complexity of linear search is in the order of n, i.e., linear in n.

Coding Demonstration

Binary Search Algorithm

- If the array `A` is sorted, then the binary search algorithm is much more efficient than linear search, which has a complexity in the order of $\log_2 n$.
- We will assume that the array `A` has been sorted in ascending (i.e., increasing) order, and that `A[0]` $\leq$ `K` $\leq$ `A[n-1]`.
- Idea: Maintain a search range for the key index, and narrow down the range by half in each iteration.
  - Try working out a couple of examples by hand
  - Use a while-loop

What is the state of the loop?

- The interval to be tested, specified in terms of its lower index (variable **indexLow**) and upper index (variable **indexHigh**)
- whether an element in the array has been found to match the key so far and its index in the array.
  $\Rightarrow$ use an int-typed variable **pos** (standing for "position") to denote the index of the found element that matches the key; it takes value -1 if the element has not been found so far

```
int indexLow=0, indexHigh=n-1, pos=-1, midIndex;
while ((indexLow<=indexHigh) && (pos==-1))
{
  midIndex=(indexLow+indexHigh)/2;
  if (K==A[midIndex])
    pos=midIndex;
  else if (K>A[midIndex])
    indexLow=midIndex+1;
  else
    indexHigh=midIndex-1;
}
```

Write a Program

Write a program that asks the user to enter 9 integers in ascending orders and saves the numbers in order into an array. The program then asks the user to enter another integer. If the integer is in the array, it prints the index of the integer in the array; otherwise it prints "not found". You must use binary search.

In-Class Exercise: