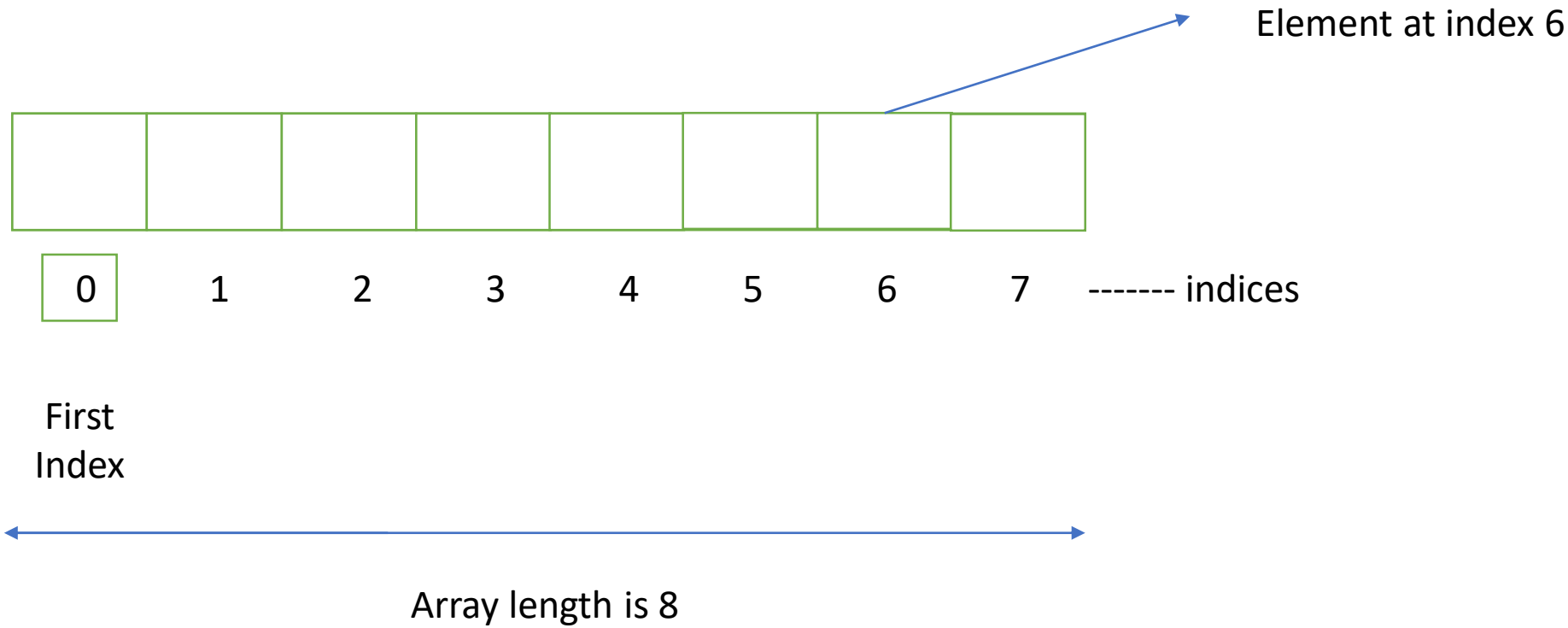
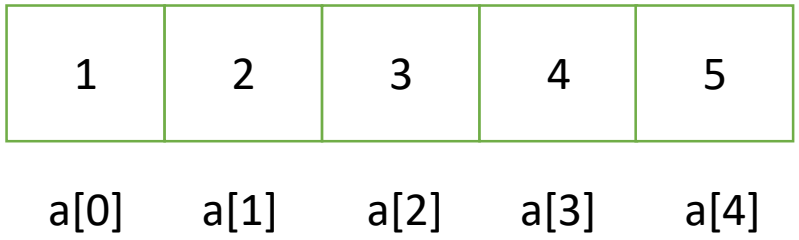
Arrays

Definition

- a large and impressive grouping/organization of things.
- regular order or arrangement; series



Declaration and Initialization



- Array in C is declared/defined similar to defining a variable.

  int a[5]; or

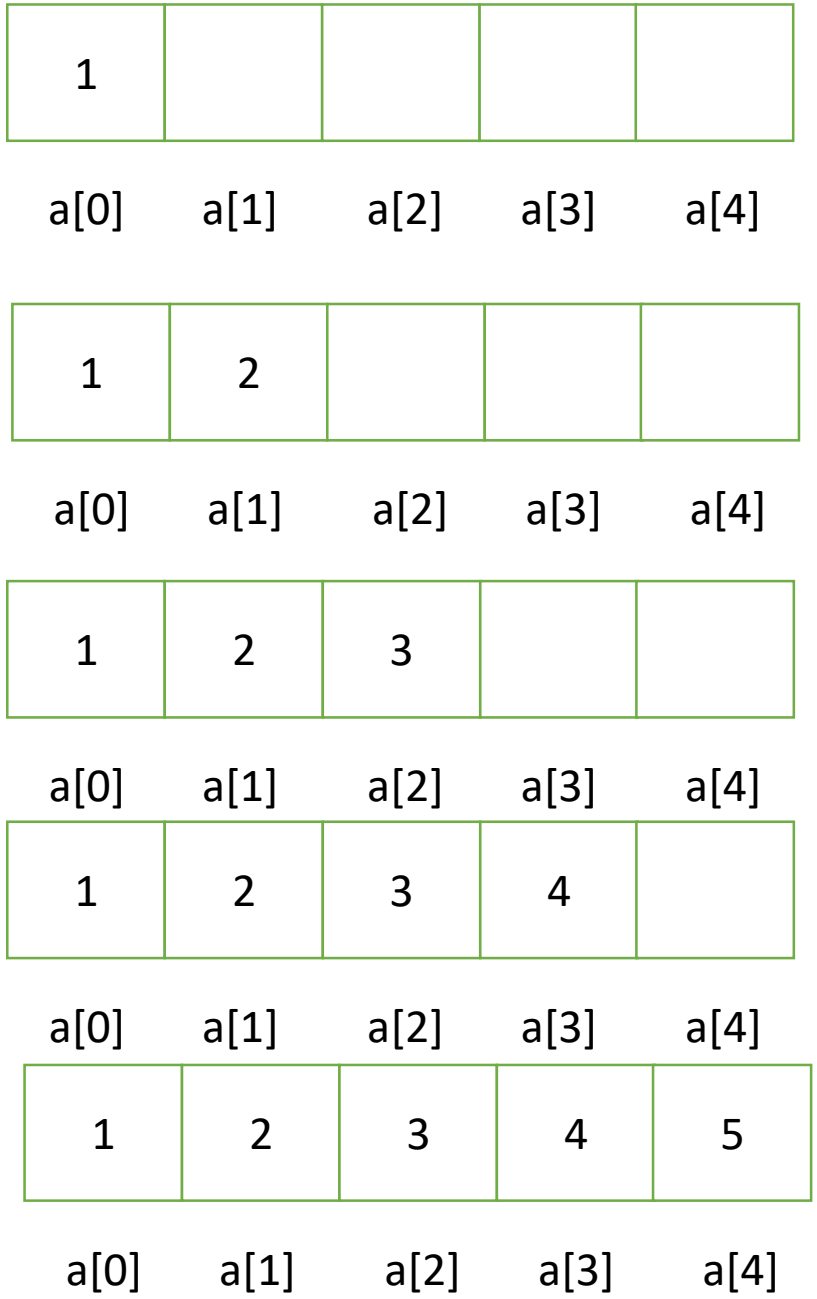
  int a[] = {1, 2, 3, 4, 5, 6, 7, 8};
- The square parenthesis indicates that a is not a single integer but an array, that is consecutively allocated group, of 5 integers.
- It creates five integer variables.
- The boxes are addressed as a[0], a[1], a[2], a[3], a[4]. These are called the elements of the array.
- If you mention size with initialization, the size much be atleast as large as number of initialized elements.

  int a[5] = {1, 2, 3}; //OK. Remaining elements will be 0.

  int a[4] = {1, 2, 3, 4, 5} //Warning

Example

```
#include<stdio.h>
void main() {
        int i;
        int a[5];
        for(i=0; i<5; i++) {
                a[i] = i+1;
                printf("%d", a[i]);
        }
}
```



Array of float/char

- One can define an array of float or an array of char, or array of any data type of C. For example:

```
void main() {
        float num[100]; //Defines array of 100 floats from 0 to 99
        num[0], num[1], num[2], ... num[99]
        char s[256]; //Defines array of 256 characters from 0 to 255
        s[0], ... s[255]
        ...
}
```

Importance of Array Size

```
void main() {
        int a[5];
        ...
}
```

- This code has 5 integers a[0], ..., a[4].
- Can you access a[5]? This is undefined. But compiler will generate a warning. However, your program may crash (segmentation fault).
- This, no compiler error, but never recommended.

User Input into an Array

```
#include<stdio.h>
void main() {
        int num[10];
        for(i=0; i<10; i++) {
                scanf("%d", &num[i]);
        }
        for(i=0; i<10; i++) {
                printf("%d\n", num[i]);
        }

}
scanf("%d", &a)
```

- To read a variable, you need the address of the variable.
- Thus, you will need &.
- Next, you need variable name.
- In array, a variable name is a combination of name and index.
- 3rd variable == num[2]
- Also, &num[i] is evaluated as &(num[i]).

Example: Reverse a line of text.

- Give the following text.

Welcome to CSVTU!

We need to print:

!UTVSC ot emocleW

```
#include<stdio.h>
void main() {
        //Define array to hold 100 characters
        //Read 100 characters from user one-by-one and store in the array
        //Print the characters in reverse
}
```

Example: Step 1

```
#include<stdio.h>
void main() {
        char ch[100];
}
```

- In next step, we can either read characters till 100 characters are read. However, if we want to stop before 100 characters, what to do then?

getchar() and EOF

- getchar() is a function to read character only from I/O stream (keyboard).
- EOF is a special character returned by getchar() when you press ctrl+D in unix or ctrl+Z in windows. It marks the end of input.

```
#include<stdio.h>
void main() {
        char ch;
        while((ch=getchar())!=EOF) {
                printf("Still Running!\n");
        }
        printf("Stopped!\n");
}
```

Example: Step 2

```
#include<stdio.h>
void main() {
        char ch[100], chr;
        int count=0;
        while((chr=getchar()) != EOF && count < 100) {
                ch[count] = chr;
                count++;
        }
}
```

- Next and final step will be to print the array in reverse.

Example: Step 3

```
#include<stdio.h>
void main() {
        char ch[100], chr;
        int count=0, i;
        while((chr=getchar()) != EOF
&& count < 100) {
                ch[count] = chr;

                count++;
        }
i = count - 1;
printf("\nReverse string is:");
while(i>=0) {
        putchar(ch[i]);
        i--;
}
}
```

Array and Parameter Passing

- You can pass individual elements of an array into a function similar to passing variables.

```
void swap(int a, int b) {
        ...
}
void main() {
        int ch[5];
        ...
        swap(ch[0], ch[1]);
        ...
}
```

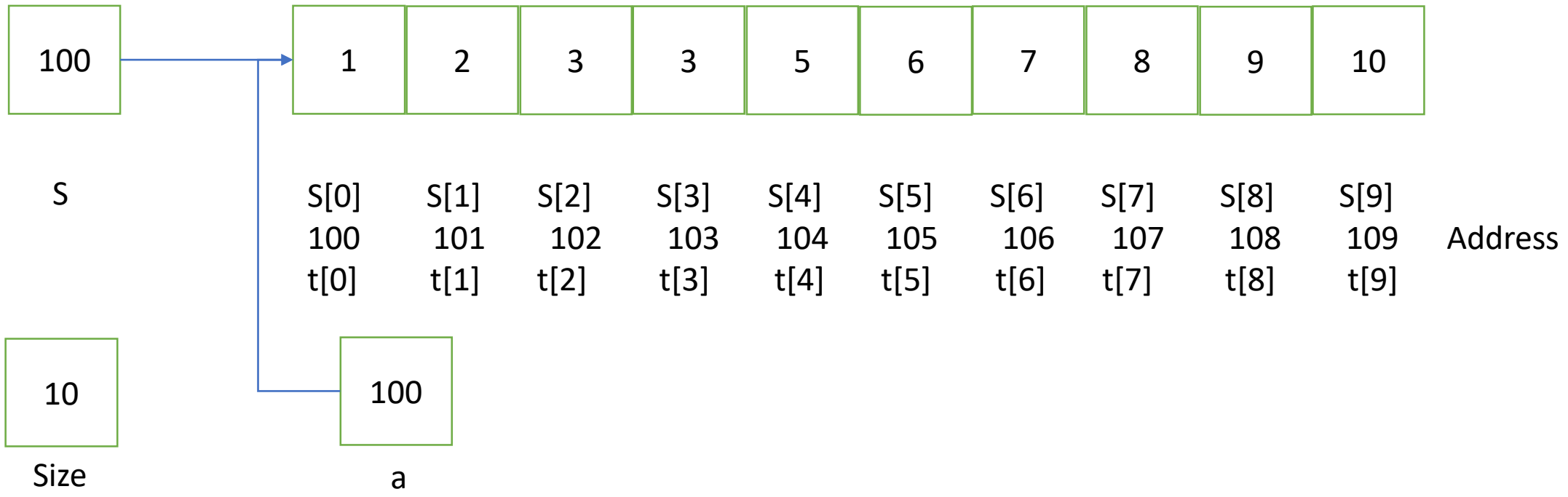
Array and Parameter Passing

- Write a function that reads input into an array of characters until EOF is seen or array is full.

```
int read_into_array(char t[], int size) {
        int ch;
        int count = 0;
        ch=getchar();
        while(count < size && ch != EOF) {
                t[count] = ch;
                count = count + 1;
                ch = getchar();
        }
        return count;
}
```

```
int main() {
        char s[10];
        read_into_array(s, 10);
        ...
}
```

Memory Representation



Example: Vector dot product

- Write a function dot_product that takes as argument two integer arrays, a and b, and an integer, size, and computes the dot product of first size elements of a and b.

- Function declaration or prototype

```
int dot_product(int a[], int b[], int size);   or
int dot_product(int [], int [], int);
```

Program: Skeleton

```
#include<stdio.h>
int dot_product(int [], int [], int);
void main() {
        int vec1[] = {2, 4, 1, 7, -5, 0, 3, 1};
        int vec2[] = {5, 7, 1, 0, -3, 8, -1, -2};
        printf("%d", dot_product(vec1, vec2, 8));
        printf("%d", dot_product(vec2, vec1, 8));
}
```

p = 0 + 2*5 = 10

p = 10 + 4*7 = 38

..

Function dot_product

$p = \sum_{i=1}^{size} (a_i * b_i)$

```
int dot_product(int a[], int b[], size) {
        int p=0, i;
        for(i=0; i<size; i++) {
                p = p + (a[i] * b[i]);
        }
        return p;
}
```

Practice Problem 1

- Read an array of 5 floats.
  - Compute the mean.
  - Print the difference of each element from the mean.

- Input: 3 1 5 2 9
- Output: -1.00 -3.00 1.00 -2.00 5.00