

Problem Solving With C++

Mohammad A. Rajabi

Dept. of Geomatics Eng.

University of Tehran

Cell: 0912 132 5823

Email: marajabi@ut.ac.ir

<http://www.marajabi.com>

Arrays

- Arrays offers convenient means of **grouping** together several related variables

One- dimensional arrays

type var_name[size]

```
int sample[10];
```

```
double d[30];
```

```
char ch[100];
```

Arrays

```
int main()
{
    int sample[10]; // this reserves 10 integer elements
    int t;
    // load the array
    for(t=0; t<10; ++t) sample[t]=t;
    // display the array
    for(t=0; t<10; ++t) cout << sample[t] << ' ';
    return 0;
} //output ?
```

Arrays

Output is:

0 1 2 3 4 5 6 7 8 9

Arrays

- In C++ all arrays are stored in **contiguous memory** locations
- An individual element is accessed by use of an index.
- An index describes the **position** of an element within an array.
- In C++ all arrays have **zero** as the index of their first element.

Arrays

- Total size of an array in bytes?
- Total bytes = number of bytes in type x number of elements
- e.g. `int sample[10];`

Arrays

```
int main()
{
    int i, min_value, max_value;
    int list[10];

    for(i=0; i<10; i++){
        list[i] = rand();
        cout<<list[i]<<" ";
    }
}
```

Arrays

```
// find minimum value
```

```
min_value = list[0];
```

```
for(i=0; i<10; i++)
```

```
    if(min_value>list[i]) min_value = list[i];
```

```
cout << "\nminimum value: " << min_value << '\n';
```

Arrays

```
// find maximum value
```

```
max_value = list[0];
```

```
for(i=0; i<10; i++)
```

```
    if(max_value<list[i]) max_value = list[i];
```

```
cout << "maximum value: " << max_value << '\n';
```

```
return 0;
```

```
}
```

Arrays

```
int a[10], b[10];
```

```
// ...
```

```
a = b; // error -- illegal
```

Arrays - No bounds checking

// An incorrect program. Do Not Execute!

```
int main()
```

```
{
```

```
    int crash[10], i;
```

```
    for(i=0; i<100; i++) crash[i]=i;
```

```
    return 1;
```

```
}
```

Sorting an array

// Using the bubble sort to order an array.

```
int main()
```

```
{  
    int nums[10];  
    int a, b, t;  
    int size;  
    size = 10; // number of elements to sort  
    // give the array some random initial values  
    for(t=0; t<size; t++) nums[t] = rand();
```

Sorting an array

```
// display original array
```

```
cout << "Original array is: ";
```

```
for(t=0; t<size; t++)
```

```
    cout << nums[t] << ' ';
```

```
cout << '\n';
```

Sorting an array

```
// This is the bubble sort.
```

```
for(a=1; a<size; a++)
```

```
for(b=size-1; b>=a; b--) {
```

```
if(nums[b-1] > nums[b]) { // if out of order
```

```
    // exchange elements
```

```
    t = nums[b-1];
```

```
    nums[b-1] = nums[b];
```

```
    nums[b] = t;
```

```
}
```

```
} // This is the end of the bubble sort.
```

Sorting an array

```
// display sorted array
```

```
cout << "Sorted array is: ";
```

```
for(t=0; t<size; t++) cout << nums[t] << ' ';
```

```
return 0;
```

```
}
```

Strings

- One dimensional arrays is used to create character strings
- In C++, a string is defined as a **character** array that is terminated by a null
- A null is specified using ‘\0’ and **is zero**
- Because of the null terminator, it is necessary to declare a character to be **one character longer** than the largest string it will hold.

Strings

- Examples:

`"hello there"`

`"I like C++"`

`""`

`"#$%@@#$"`

Reading strings from keyboard

- The easiest way to read a string entered from the keyboard is to make a **character** array

```
int main()
{
    char str[80];
    cout << "Enter a string: ";
    cin >> str; // read string from keyboard
    cout << "Here is your string: ";
    cout << str<<"\n";

    return 0;
```

```
}
```

Reading strings from keyboard

- There is a problem with previous program, if the string has whitespace characters

// Using gets() to read a string from keyboard. (stdio.h)

```
int main()
{
    char str[80];
    cout << "Enter a string: ";
    gets(str); // read a string from the keyboard
    cout << "Here is your string: ";
    cout << str<<"\n";

    return 0;
}
```

Reading strings from keyboard

- Remember neither `cin` nor `gets()` performs **any bounds checking** on the array

String Library Functions

strcpy() strcat() strlen() strcmp()

//concatenate, header file: string.h

strcpy(to, from)

```
int main()
{
    char str[80];
    strcpy(str, "hello");
    cout << str;
    return 0;
}
```

Output is: **hello**

String Library Functions (**strcat**)

```
int main()
{
    char s1[20], s2[10];
    strcpy(s1, "hello");
    strcpy(s2, " there");
    strcat(s1, s2);
    cout << s1<<<“\n”;
    return 0;
} //output?
```

String Library Functions (**strcmp**)

- **strcmp(s1,s2)** compares two strings and **returns 0** if they are equal. If s1 is greater than s2(dictionary order) then a positive number is returned; if it is less than s2 then a negative number is returned.

```
bool password();
```

```
int main()
```

```
{
```

```
    if(password()) cout << "Logged on.\n";
```

```
    else cout << "Access denied.\n";
```

```
    return 0;
```

```
} Mohammad A. Rajabi
```

Problem Solving With C++

String Library Functions (**strcmp**)

// Return true if password accepted; false otherwise.

```
bool password()
```

```
{
```

```
char s[80];
```

```
cout << "Enter password: ";
```

```
gets(s);
```

```
if(strcmp(s, "password")) { // strings differ
```

```
    cout << "Invalid password.\n";
```

```
    return false;
```

```
}
```

```
// strings compared the same
```

```
return true;
```

Write a program to continuously
input a string from the user until
the user types quit

String Library Functions (strcmp)

```
int main()
{
    char s[80];
    for(;;) {
        cout << "Enter a string: ";
        gets(s);
        if(!strcmp("quit", s)) break;
    }
    return 0;
}
```

String Library Functions (**strlen**)

strlen(s) returns the length of s

```
int main()
{
    char str[80];
    cout << "Enter a string: ";
    gets(str);
    cout << "Length is: " << strlen(str)<<"\n";
    return 0;
}
```

//what is the output if we enter: this is cs192

Write a program to input a string
from the user and print it
backwards (use strlen)

String Library Functions (strlen)

```
#include <iostream>
#include <cstdio> // for older compilers, use <stdio.h>
#include <cstring> // for older compilers, use
    <string.h>
using namespace std;
int main()
{
    char str[80];
    int i;
    cout << "Enter a string: ";
    gets(str);
    for(i=strlen(str)-1; i>=0; i--) cout << str[i];
    return 0;
}
```

Example: String Functions

```
int main()
{
    char s1[80], s2[80];
    cout << "Enter two strings: ";
    gets(s1); gets(s2);
    cout << "lengths: " << strlen(s1);
    cout << ' ' << strlen(s2) << '\n';
    if(!strcmp(s1, s2))
        cout << "The strings are equal\n";
    else cout << "not equal\n";
    strcat(s1, s2);
    cout << s1 << '\n';
    strcpy(s1, s2);
    cout << s1 << " and " << s2 << ' ';
    cout << "are now the same\n";
    return 0;
}
```

Using the Null Terminator

```
int main()
{
    char str[80];
    int i;
    strcpy(str, "this is a test");
    for(i=0; str[i]; i++)
        str[i] = toupper(str[i]);
    cout << str << endl;
    return 0;
}
```

Using Null Terminator

- Test condition of the for loop is simply the array indexed by `i`
- loop runs until it encounters the **null terminator**.

- `toupper()` `tolower()` `isalpha()`
- `isdigit()` `isspace()` `ispunct()`

- Header file: `stdlib.h` and `ctype.h`

Two Dimensional Arrays

- `int twod[10][20];`

```
int main()
{
    int t,i, num[3][4];
    for(t=0; t<3; ++t) {
        for(i=0; i<4; ++i) {
            num[t][i] = (t*4)+i+1;
            cout << num[t][i] << ' ';
        }
        cout << '\n';
    }
    return 0;
```

Two Dimensional Arrays

- **What happens when I change num[3][4] to num[2][6]?**

```
int main()
{
    int t,i, num[2][6];
    for(t=0; t<2; ++t) {
        for(i=0; i<6; ++i) {
            num[t][i] = (t*4)+i+1;
            cout << num[t][i] << ' ';
        }
        cout << '\n';
    }
    return 0;
```

Two Dimensional Arrays

- **Two dimensional arrays are accessed in a row-column matrix.**
- **First index indicates the row and the second indicates the column**
- **bytes = row x column x number of bytes in type**

Multidimensional Arrays

- `type name[size1][size2] . . . [sizeN];`
- `int multidim[4][10][3];`
- when multidimensional arrays are used, large amounts of memory are consumed.

Array Initialization

- `type array_name[size] = [value-list];`
- `int i[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};`
- Character arrays allow a shorthand initialization
- `char array_name[size] = "string";`
- `char str[6] = "hello";` // why is size 6?
- `char str[6] = {'h', 'e', 'l', 'l', 'o', '\0'};`

Array Initialization

- `char str[6] = "hello";`
- compiler automatically supplies the null terminator.

Multidimensional Array Initialization

- `int sqrs[10][2] = {`

`1, 1,`

`2, 4,`

`3, 9,`

`4, 16,`

`5, 25,`

`6, 36,`

`7, 49,`

`8, 64,`

`9, 81,`

`10, 100`

`Mohammad A. Rajabi`

`Problem Solving With C++`

`};`

Multidimensional Arrays

```
int sqrs[10][2] = {  
    {1, 1},  
    {2, 4},  
    {3, 9},  
    {4, 16},  
    {5, 25},  
    {6, 36},  
    {7, 49},  
    {8, 64},  
    {9, 81},  
    {10, 100}  
};
```

Multidimensional Arrays

```
int main()
{
    int i, j;
    cout << "Enter a number between 1 and 10: ";
    cin >> i;
    // look up i
    for(j=0; j<10; j++)
        if(sqrs[j][0]==i) break;
    cout << "The square of " << i << " is ";
    cout << sqrs[j][1];
    return 0;
}
```

Arrays

```
void f1();
```

```
int main()
```

```
{
```

```
    f1();
```

```
    f1();
```

```
    return 0;
```

```
}
```

```
void f1()
```

```
{
```

```
    char s[80]="this is a test\n";
```

```
    cout << s;
```

```
    strcpy(s, "CHANGED\n"); // change s
```

```
    cout << s;
```

```
} Mohammad A. Rajabi
```

```
Problem Solving With C++
```

Arrays

- **Global arrays** are initialized when the program begins. **Local arrays** are initialized each time the function that contains them is called.

Unsize Arrays Initializations

- `char e1[14] = "Divide by 0\n";`
- `char e2[23] = "End-of-File\n";`
- `char e3[21] = "Access Denied\n";`

- `char e1[] = "Divide by 0\n";`
- `char e2[] = "End-of-File\n";`
- `char e3[] = "Access Denied\n";`

Unsize Arrays Initializations

```
int sqrs[][2] = {
```

```
    1, 1,
```

```
    2, 4,
```

```
    3, 9,
```

```
    4, 16,
```

```
    5, 25,
```

```
    6, 36,
```

```
    7, 49,
```

```
    8, 64,
```

```
    9, 81,
```

```
    10, 100
```

```
};
```

```
};
```

Problem Solving With C++

Arrays of Strings

- The following declares an array of 30 strings, each having a maximum length of 80 characters
- `char str_array[30][80];`
- In order to access an individual string, specify only the left index.

Example:

- `gets(str_array[2]); /* calls gets with third string in str_array*/`

Arrays of Strings

```
int main()
{
    int t, i;
    char text[100][80];
    for(t=0; t<100; t++) {
        cout << t << ": ";
        gets(text[t]);
        if(!text[t][0]) break; // quit on blank line
    }
    // redisplay the strings
    for(i=0; i<t; i++)
        cout << text[i] << '\n';

    return 0;
}
}
}
```

Example using String Arrays

```
char name[10][80]; // this array holds employee names
char phone[10][20]; // their phone numbers
float hours[10]; // hours worked per week
float wage[10]; // wage
```

Example using String Arrays

```
void enter() // Enter information.  
{  
    int i;  
    for(i=0; i<10; i++) {  
        cout << "Enter last name: ";  
        cin >> name[i];  
        cout << "Enter phone number: ";  
        cin >> phone[i];  
        cout << "Enter number of hours worked: ";  
        cin >> hours[i];  
        cout << "Enter wage: ";  
        cin >> wage[i];  
    }  
}
```

Example using String Arrays

```
// Display report.
```

```
void report()
```

```
{
```

```
    int i;
```

```
    for(i=0; i<10; i++) {
```

```
        cout << name[i] << ' ' << phone[i] << '\n';
```

```
        cout << "Pay for the week: " << wage[i] * hours[i];
```

```
        cout << '\n';
```

```
    }
```

```
} //employee example
```

Example

```
int main()
{
    char city[80]="Islamabad";
    char places[4][80]={"Lahore", "Hanoi", "Karachi", "Madrid"};
    cout << places[3] << '\n';
    cout <<places[2][4]<<'\n';
    cout <<places[1][6]<<'\n';
    cout<<city<<'\n';
    cout<<city[4]<<'\n';
    return 0;
}
```