7.2 Array Basics

Monday, February 13, 2023 10:30 AM

- Array- to store collection of data, collection of var of same type
- Declare one array w/ name (like 'number'), and the individual vars with [] (like 'number[1]')

7.2.1 Declaring Arrays

Monday, February 13, 2023 10:33 AM

- ElementType arrayName[SIZE];
- ElementType can be any data type, all elements in array would have same type
- SIZE aka array size declarator, is expression to int > 0
- To set var to certain val,
  - ArrayName[index] = value;
-

```
#include <iostream>
using namespace std;

int main()
{
  int x[3];
  cout << "x[0] is " << x[0];

  return 0;
}
```

-

- The program has a compile error because the size of the array wasn't specified when declaring the array.
- The program has a runtime error because the array elements are not initialized.
- The program runs fine and displays x[0] is 0.
- The program has a runtime error because the array element x[0] is not defined.
- ✓ x[0] has an arbitrary value.

-

7.2.2 Accessing Array Elements

Monday, February 13, 2023 10:43 AM

- Array indices- go from 0 to arraySize-1
- MyList[0]++; would increment the element by 1
- Cout<<max(myList[1], myList[2])<<endl; will print out the max number btwn the two

7.2.3 Array Initializers

Monday, February 13, 2023 10:55 AM

- Short hand notation, array initializer
- ```
  elementType arrayName[arraySize] = {value0, value1, ..., valuek};
  ```
- Can also just do it normally, which is like:
- ```
  double myList[4];
  myList[0] = 1.9;
  myList[1] = 2.9;
  myList[2] = 3.4;
  myList[3] = 3.5;
  ```
- But don't do this:
- ```
  double myList[4];
  myList = {1.9, 2.9, 3.4, 3.5};
  ```
- Can also omit the array size when declaring & creating array using an initializer, compiler auto figures out how many elements are in the array
- Can also initialize part of an array, the other parts would get 1
-
  ```

7.2.4 Processing Arrays

Monday, February 13, 2023 11:56 AM

- Usually use for loop
- 10 ex of processing arrays
  1. Initializing arrays w/ input vals:
     i.
```
cout << "Enter " << ARRAY_SIZE << " values: ";
for (int i = 0; i < ARRAY_SIZE; i++)
  cin >> myList[i];
```
  2. Initializing arrays w/ rand vals
     i.
```
for (int i = 0; i < ARRAY_SIZE; i++)
{
  myList[i] = rand() % 100;
}
```
  3. Printing Arrays:
     i.
```
for (int i = 0; i < ARRAY_SIZE; i++)
{
  cout << myList[i] << " ";
}
```
  4. Copying arrays:
     i.
```
list = myList;
```
     ii. This is wrong, it should be like this:
     iii.
```
for (int i = 0; i < ARRAY_SIZE; i++)
{
  list[i] = myList[i];
}
```
  5. Summing all Elements:
     i.
```
double total = 0;
for (int i = 0; i < ARRAY_SIZE; i++)
{
  total += myList[i];
}
```
  6. Finding largest element:

i.

```
double max = myList[0];
for (int i = 1; i < ARRAY_SIZE; i++)
{
  if (myList[i] > max) max = myList[i];
}
```

7. Finding smallest index of the largest element:

i.

```
double max = myList[0];
int indexOfMax = 0;
for (int i = 1; i < ARRAY_SIZE; i++)
{
  if (myList[i] > max)
  {
    max = myList[i];
    indexOfMax = i;
  }
}
```

8. Random Shuffling:

i.

```
srand(time(0));
double myList[] = {1, 2, 3, 4, 5, 6};

for (int i = 0; i < ARRAY_SIZE; i++)
{
  // Generate an index j randomly
  int j = rand() % ARRAY_SIZE;

  // Swap myList[i] with myList[j]
  double temp = myList[i];
  myList[i] = myList[j];
  myList[j] = temp;
}
```

9. Shifting Elements:

i.

```
double myList[] = {4, 5, 6, 7, 8, 9};
double temp = myList[0]; // Retain the first element

// Shift elements left
for (int i = 1; i < ARRAY_SIZE; i++)
{
  myList[i - 1] = myList[i];
}

// Move the first element to fill in the last position
myList[ARRAY_SIZE - 1] = temp;
```

10. Simplifying Coding:

```
string months[] = {"January", "February", ..., "December"};
cout << "Enter a month number (1 to 12): ";
int monthNumber;
cin >> monthNumber;
cout << "The month is " << months[monthNumber - 1] << endl;
```

i. If you didn't use the **months** array, you would have to det month name using a lengthy multiway **if-else** statemen follows:

```
if (monthNumber == 1)
  cout << "The month is January" << endl;
else if (monthNumber == 2)
  cout << "The month is February" << endl;
...
else
  cout << "The month is December" << endl;
```

11.
12.

7.2.5 Foreach Loops

Monday, February 13, 2023 6:28 PM

- For(double e: myList){…}
- Can traverse array sequentially w/out using index var, so this ^ goes thru array myList
- "for each element e in myList, do ..."
-
```
numberOfIncompletes = 0;
for(k=0;k<nIncompletes;k++){
    if(incompletes[k] == studentId){
        numberOfIncompletes++;
    }
}
```
-

7.3 Case Study: Analyzing Numbers

Tuesday, February 14, 2023 1:57 PM

- Pgrm to find numb of items above avg of items
-

```
#include <iostream>
using namespace std;

int main()
{
  const int NUMBER_OF_ELEMENTS = 3; // For simplicity, use 3 (not 100) for demo
  double numbers[NUMBER_OF_ELEMENTS];
  double sum = 0;

  for (int i = 0; i < NUMBER_OF_ELEMENTS; i++)
  {
    cout << "Enter a new number: ";
    cin >> numbers[i];
    sum += numbers[i];
  }

  double average = sum / NUMBER_OF_ELEMENTS;

  int count = 0; // The number of elements above average
  for (int i = 0; i < NUMBER_OF_ELEMENTS; i++)
    if (numbers[i] > average)
      count++;

  cout << "Average is " << average << endl;
  cout << "Number of elements above the average " << count << endl;

  return 0;
}
```

-

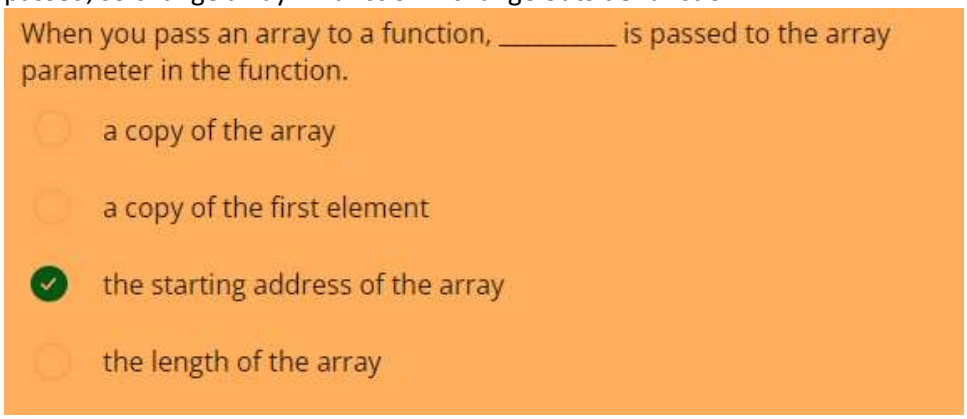
7.4 Case Study: Deck of Cards

Tuesday, February 14, 2023 2:04 PM

- Pgrm tha rando select 4 cards from deck of 52
- All cards can be repped in array (called deck) with intial vals 0-51
  - 0-12 = 13 spades
  - 13-25 = 13 hearts
  - 26-38 = 13 diamonds
  - 39-51 = 13 clubs
- Cardnumber % 13 = rank of card
- Use if-else to check the /,
  - If == 0 : spades
  - If == 1 : heart
  - If == 2 : diamonds
  - If == 3 : clubs
-

7.5 Passing Arrays to Functions

Wednesday, February 15, 2023 10:27 AM

- Can pass entire array to a function
- Just put it in the parameter in the function header, written as (int list[]) (diff than java)
- Name of parameter can be omitted
- ```
  void printArray(int [], int); // Function prototype
  ```
- Usually when passing in an array, pass in the size as well to know how many elements in it
- C++ uses pass-by-value to pass array args to a function, diff btwn pass-by-val of vars of primitive types and arrays
  - Primitive- argument's value is passed
  - Array- val of arg is the starting mem address to an array, val is passed to array parameter in function, so best described as pass-by-sharing, like array in function same as array being passed, so change array in function = change outside function
  - 

  -
  ```

7.6 Preventing Changes of Array Arguments in Functions

Wednesday, February 15, 2023 10:44 AM

- Can define const array parameter in function so that array don't get changed
- Bc when passing array, all that sent is starting mem address of array
- Use 'const' to not mod the array
- If go to 2 methods:

Note

If you define a `const` parameter in a function `f1` and this parameter is passed to another function `f2`, then the corresponding parameter in function `f2` must be declared `const` for consistency. Consider the following code:

-
```
void f2(int list[], int size)
{
  // Do something
}

void f1(const int list[], int size)
{
  // Do something
  f2(list, size);
}
```

The compiler reports an error, because `list` is `const` in `f1` and it is passed to `f2`, but it is not `const` in `f2`. The function declaration for `f2` should be

-
```
void f2(const int list[], int size)
```

-

7.7 Returning Arrays from Functions

Wednesday, February 15, 2023 11:09 AM

- To return array from a function, pass it as a parameter in a function
- Instead of passing in one array and then trying to return the array (it wont work, that's illegal), just pass In 2 arrays, one gets modified
- Just pass in 1 more array than actually using in the function

7.8 Case Study: Counting the occurrences of Each Letter

Thursday, February 16, 2023 1:47 PM

- Pgrm to count occurrences of each letter in an array of characters
- Pgrm makes 100 lowercase letters rando and assign them to array of chars
  - ```
    static_cast<char>('a' + rand() % ('z' - 'a' + 1))
    ```
- Count occurrences of each letter in the array by making array of 26 int
- Now do for loop that basically auto converts the char in the og array to num, then subtract 'a', and that number will be the array slot in the new array we made, we will increment that slot.
  - So if chars is the array with all the rando letters, and counts is the array with 26 slots (0-25), then:
  - ```
    for (int i = 0; i < NUMBER_OF_RANDOM_LETTERS; i++)
      counts[chars[i] - 'a']++;
    ```
  -

7.9 Searching Arrays

Thursday, February 16, 2023 1:57 PM

- If array is sorted, binary search more efficient than linear search for finding an element in the array
- Search for looking for specific element in array
- Linear search approach
  - Compares key element key sequentially w/ each element in array, if match, returns index of element, otherwise returns –1
- Binary search approach
  - Assume that elements in array already ordered, in ascending order
  - First compares key w/ middle element, then if key less than (or greater than), only have to check half of the array
  - Repeat w/ loop
  -

```
int binarySearch(const int[] list, int key, int listSize)
{
  int low = 0;
  int high = listSize - 1;

  while (high >= low)
  {
    int mid = (low + high) / 2;
    if (key < list[mid])
      high = mid - 1;
    else if (key == list[mid]
      return mid;
    else
      low = mid + 1;
  }

  return -1; // Not found
}
```

Version 2 adds a loop to search for the key repeatedly. The added code is highlighted in red.

  - When key not found, low is the insertion point (index where a key would be inserted to keep the order of the list)
  - When not in array, return –1 bc key is less than list[0], and can't return 0 (that is an actual pos), so can return –low-1
- Properties of functions usually defined by pre & post conditions
  - For this case, the pre would be list must be in increasing order, post would be that function returns index of element that matches the key or negative integer (if its not in the list)

For the binarySearch function in this section, what is low and high after the first iteration of the while loop for the list {1, 4, 6, 8, 10, 15, 20} and key 11?

- low is 0 and high is 6
- low is 0 and high is 3
- low is 3 and high is 6
- low is 4 and high is 6
- low is 0 and high is 5

7.10 Sorting Arrays

Thursday, February 16, 2023 2:18 PM

- Sorting, like searching, is common, many algorithms dev, we do selection sort
- Selection sort finds smallest numb in list and swaps with first, then next smallest and swap with seconds, …
-

```
#include <iostream>
using namespace std;

void selectionSort(double [], int);

int main()
{
  double list[] = {-2, 4.5, 5, 1, 2, -3.3};
  selectionSort(list, 6);

  for (int i = 0; i < 6; i++)
  {
    cout << list[i] <<  " ";
  }

  return 0;
}

void selectionSort(double list[], int listSize)
{
  for (int i = 0; i < listSize - 1; i++)
  {
    // Find the minimum in the list[i..listSize-1]
    double currentMin = list[i];
    int currentMinIndex = i;

    for (int j = i + 1; j < listSize; j++)
    {
      if (currentMin > list[j])
      {
        currentMin = list[j];
        currentMinIndex = j;
      }
    }

    // Swap list[i] with list[currentMinIndex] if necessary;
    if (currentMinIndex != i)
    {
      list[currentMinIndex] = list[i];
      list[i] = currentMin;
    }
  }
}
```

- The selectionSort function sorts array of double elements, dun in nested for loop
-

7.11 C-Strings

Thursday, February 16, 2023 2:29 PM

- C-string is an array of characters that ends with the null terminator character '\0' , you can process C-strings using C-strings usings C-string functions in the C++ library
- string is much easier and better, but this lets us mess w/ arrays
- \ is escape sequence
- Last character of the array is the null terminator
- Each character is one spot of the array
-

Are the following two declarations the same?

```
char city[] = {'D', 'a', 'l', 'l', 'a', 's'};
char city[] = "Dallas";
```

- yes
- no

That's incorrect.

No. The first one is an array of characters. The second one is a C-String with a terminating character at the end.

-

7.11.1 Input and Output of C-Strings

Thursday, February 16, 2023 2:35 PM

- Output C-string is simple, if s is array for a C-string, then just cout<<s;
- When read string to an array, leave room for null terminator char, but can't read whitespace char, so can't read in spaced words
- Just use cin.getline function in iostream header file, reads a string into an array
- ```
  cin.getline(char array[], int size, char delimitChar)
  ```
- Function stops reading after delimiter char or when size-1 numb of chars read
-
-
  ```

7.11.2 C-String Functions

Thursday, February 16, 2023 2:53 PM

- Bc C-string end w/ null terminator, can use this
- Don't have to pass in length of it, just count all chars from left to right in array till null term char reached
- strlen and other functions in C++ library for processing C-strings

`size_t strlen(const char s[])`	Returns the length of the string, i.e., the number of the characters before the null terminator.
`strcpy(char s1[], const char s2[])`	Copies string `s2` to string `s1`.
`strncpy(char s1[], const char s2[], size_t n)`	Copies the first `n` characters from string `s2` to string `s1`.
`strcat(char s1[], const char s2[])`	Appends string `s2` to `s1`.
`strncat(char s1[], const char s2[], size_t n)`	Appends the first `n` characters from string `s2` to `s1`.
`int strcmp(char s1[], const char s2[])`	Returns a value greater than `0`, `0`, or less than `0` if `s1` is greater than, equal to, or less than `s2` based on the numeric code of the characters.
`int strncmp(char s1[], const char s2[], size_t n)`	Same as `strcmp`, but compares up to `n` number of characters in `s1` with those in `s2`.
`int atoi(char s[])`	Returns an `int` value for the string.
`double atof(char s[])`	Returns a `double` value for the string.
`long atol(char s[])`	Returns a `long` value for the string.
`void itoa(int value, char s[], int radix)`	Obtains an `int` value to a string based on specified radix.

`size_t` is a C++ type. For most compilers, it is the same as `unsigned int`.

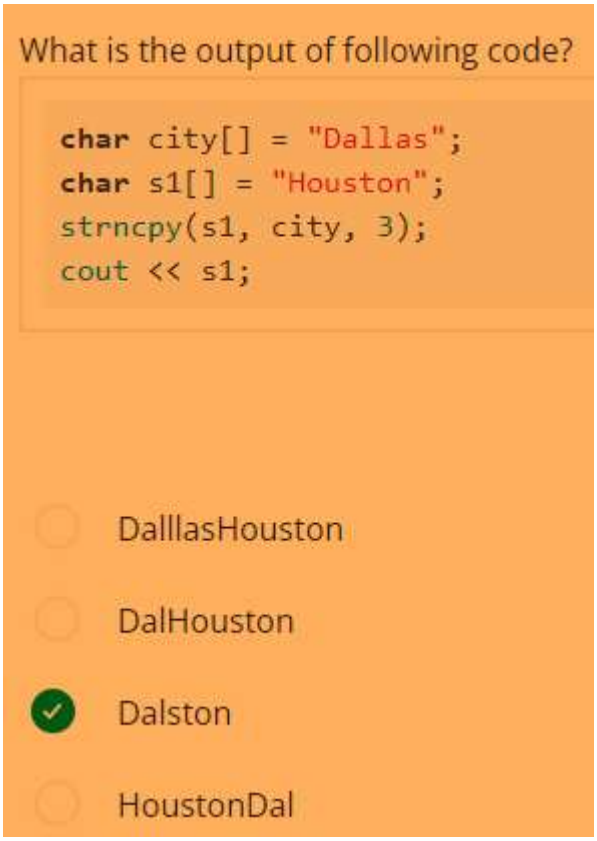
All these functions are defined in the `cstring` header file except that conversion functions `atoi`, `atof`, `atol`, and `itoa` are defined in the `cstdlib` function.

- In Visual C++ 2013 or higher, `strcpy`, `strncpy`, `strcat`, and `strncat` are replaced by `strcpy_s`, `strncpy_s`, `strcat_s`, and `strncat_s` and `itoa` is replaced by `_itoa_s`. `strcpy`, `strncpy`, `strcat`, `strncat`, and `itoa` are unsafe. When you try to copy/concat a string using `strcpy/strncpy/strcat/strncat` to a buffer which is not large enough to contain the result, it will cause a buffer overflow. `strcpy_s`, `strncpy_s`, `strcat_s`, `strncat_s`, and `_itoa_s` are safe versons. With these new versions, you can optionally specify the size of the destination buffer to avoid buffer overflows.

7.11.3 Copying Strings Using strcpy and strncpy

Thursday, February 16, 2023 3:03 PM

- strcpy used to copy a src string in the second argument to a target string in the first argument, target string must already been allocated sufficient mem for the function to work
- strcpy(c-string, "thing");
- strncpy works like strcpy, but takes 3rd argument specifying numb of chars to be copied
- If specified numb of chars is greater than length of the src string, src string copied to the target padded w/ null terminators all the way up to the end of the target string
- Both of them can potentially override the bounds of an array, so make sure to check bounds before use function
-



-

7.11.4 Concatenating Strings Using

Thursday, February 16, 2023 3:27 PM

- Function strcat can be used to append the string in second arg to string in first arg, but make sure first string has enough mem allocated
- ```
  char s1[7] = "abc";
  char s2[4] = "def";
  strcat(s1, s2);
  cout << s1 << endl; // The printout is abcdef
  ```
- strncat works like strcat, but third arg specifying the numb of chars to be concatenated from target string w/ the src string
- ```
  char s[9] = "abc";
  strncat(s, "ABCDEF", 3);
  cout << s << endl; // The printout is abcABC
  ```
- ```
  char s2[7] = "Dallas";
  char s1[14] = "Dallas";
  strcat(s1, s2);
  cout << s1;
  ```
- Dallas
  DallasDallas
  D
  DD
-
  ```

7.11.5 & .6 Comparing Strings & Converting btwn Strings to C-Strings

Thursday, February 16, 2023 3:44 PM

- strcmp can be used to compare 2 strings by corresponding chars according to their numeric codes
- Most compilers use ASCII code for chars, functions return val 0 is s1 == s2, less than 0 if s1<s2, greater than 0 if s1>s2
- s2. For example, suppose s1 is "abc" and s2 is "abg", and strcmp(s1, s2) returns a negative value. The first two characters (a vs. a) from s1 and s2 are compared. Because they are equal, the second two characters (b vs. b) are compared. Because they are also equal, the third two characters (c vs. g) are compared. Since the character c is 4 less than g, the comparison returns a negative value. Exactly what value is returned depends on the compiler. Visual C++ and GNU compilers return -1, but Borland C++ compiler returns -4 since the character c is 4 less than g.
- strncmp like strcmp, but 3rd arg, takes in numb of chars to be compared
-
-
- Usually need to convert a string object to a C-string , to convert string object s to a c-String , use s.c_str() , it returns a C-String
- To convert C-string cstr into a string, use string(cstr) , which makes string object from the C-String

7.11.7 Converting Strings to Numbers

Thursday, February 16, 2023 3:54 PM

- Function atoi can be used to convert a C-string into an int
- Function atol can be used to convert a C-string into an int of long
- Function atof can be used to convert a C-string into a floating-pt numb

7.12 Converting Numbers to Strings

Thursday, February 16, 2023 6:07 PM

- Function to_string function in the <string> header file can be used to convert a numeric val of any type to string
- It used to convert a number of any type to string object
- For converting floating-point value, trailing zeros added to resulting string so 6 digits after decimal, to get rid of trailing, can use stringstream
- ```
  int x = 15;
  double y = 1.32;
  long long int z = 10935;
  string s = "Three numbers: " + to_string(x) + ", " +
    to_string(y) + ", and " + to_string(z);
  cout << s << endl;
  ```

  displays

  ```
  Three numbers: 15, 1.320000, and 10935
  ```
- Can use function ito in <cstring> header file to convert integer to C-string based on a specified radix
  ```

```
char s1[15];
char s2[15];
char s3[15];
itoa (100, s1, 16);
itoa (100, s2, 2);
itoa (100, s3, 10);
cout << "The hex number for 100 is " << s1 << endl;
cout << "The binary number for 100 is " << s2 << endl;
cout << "s3 is " << s3 << endl;
```

- displays

```
The hex number for 100 is 64
The binary number for 100 is 1100100
s3 is 100
```

Ch 7 Summary

Thursday, February 16, 2023 6:30 PM

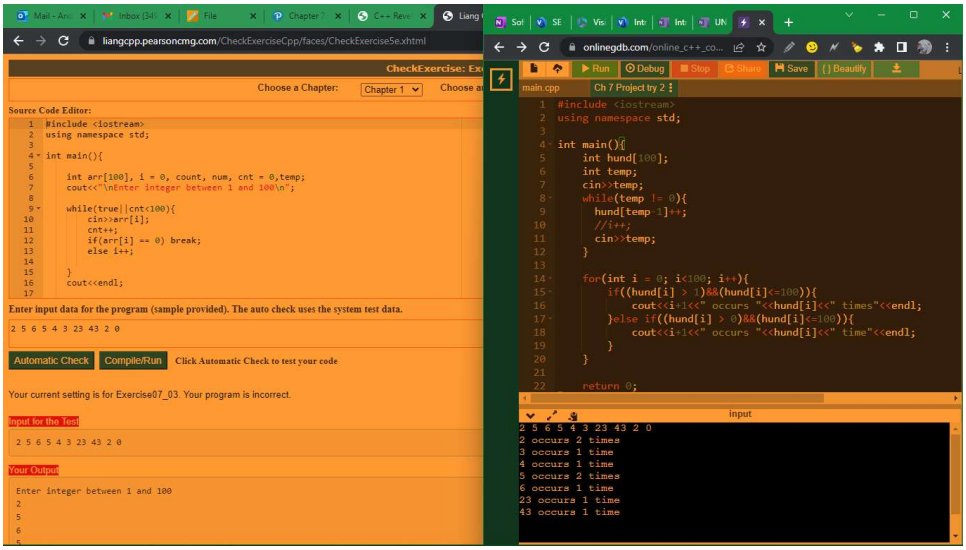
1. An array stores a list of value of the same type.
2. An array is declared using the syntax
   a. `elementType arrayName[size]`
3. Each element in the array is represented using the syntax arrayName[index].
4. An index must be an integer or an integer expression.
5. Array index is 0-based, meaning that the index for the first element is 0.
6. Programmers often mistakenly reference the first element in an array with index 1 rather than 0. This causes the index off-by-one error.
7. Accessing array elements using indexes beyond the boundaries causes out-of-bounds error.
8. Out of bounds is a serious error, but it is not checked automatically by the C++ compiler.
9. C++ has a shorthand notation, known as the array initializer, which declares and initializes an array in a single statement using the syntax:
   a. `elementType arrayName[] = {value0, value1, ..., valuek};`
10. When an array is passed to a function, the starting address of the array is passed to the array parameter in the function.
11. When you pass an array argument to a function, often you also should pass the size in another argument, so the function knows how many elements are in the array.
12. You can specify const array parameters to prevent arrays from being accidentally modified.
13. An array of characters that ends with a null terminator is called a C-string.
14. A string literal is a C-string.
15. C++ provides several functions for processing C-strings.
16. You can obtain a C-string length using the strlen function.
17. You can copy a C-string to another C-string using the strcpy function.
18. You can compare two C-strings using the strcmp function.
19. You can use the itoa function to convert an integer to a C-string, and use atoi to convert a string to an integer.
20.
21.

```cpp
#include <iostream>
using namespace std;

int main(){
    int hund[100];
    int temp;
    cin>>temp;
    while(temp != 0){
      hund[temp-1]++;
      //i++;
      cin>>temp;
    }

    for(int k = 0; k<100; k++){
        if(hund[k] > 1){
            cout<<k+1<<" occurs "<<hund[k]<<" times"<<endl;
        }else if(hund[k] > 0){
            cout<<k+1<<" occurs "<<hund[k]<<" time"<<endl;
        }
    }

    return 0;
```

```
    }

    return 0;
}
```

22.

23.



24.