Example

```
// Read in 5 scores and show how much each score differs from the
// average  score.
#include <iostream.h>

void  main( )  {
   int i,  score[5], sum=0;
   float average;

   cout << "Enter 5 scores:\n";

   for (i = 0;  i < 5;  i++)  {
        cin >> score[i];
        sum = sum + score[i];
   }
   average = float(sum) / 5 ;

   cout << "The average  score is " << max << endl
          << "The differences from average are:\n";

   for (i = 0;  i < 5;  i++)
      cout <<  score[i] <<  " off by "  <<  (average - score[i])  << endl;

   return;
}
```

Sample output:

Enter 5 scores:
5 9 3 10 8
The average score is 7
The differences from the highest are:
5 off by 2
9 off by -2
3 off by 4
10 off by -3
8 off by 1

Linear Search

Problem Definition:

Search an *ordered* array of integers for a value and return its index if the value is found. Otherwise, return -1.

0	1	2	3	4	5	6	7
7	9	10	17	30	37	46	60

Solution Outline:

```
Start with the first array element (index 0).

While array element is less than value
      and more array element do
  Try next element (increment index);

If value not found then
  return -1
else
  return index
```

Linear Search (Cont'd)

```
// Searches an ordered array of integers
int lsearch(int data[], // input: array
            int size,   // input: array size
            int value   // input: value to find
            )    // output: index if found
                 //         otherwise return -1
{
     int index = 0;    // initialize index to 0

     while ((data[index] < value) &&
            (index < size))
          index++;

     if (data[index] != value)
          return -1;
     else
          return index;
} // search


#include <iostream.h>
int main() {
     int A[10] = { 2,2,3,4,5,5,6,9,18,25 };
     int x;

     cin >> x;
     cout << lsearch(A,10,x) << endl;

     return 0;
}
```

Example

```
//Reads quiz scores for each student into the two-dimensional array
//Computes the average score for each student and the average score for
// each quiz. Displays the quiz scores and the averages.

#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>

// Global constant NO_STUDENTS and NO_QUIZZES are the
// dimensions of the array grade
const int NO_STUDENTS = 4,  NO_QUIZZES = 3;


void  std_ave   (const int grade[ ][NO_QUIZZES], double st_ave[ ]);
//Pre:    Each of the indexed variable grade[st_num-1, quiz_num-1]
//        contains the score for student st_num on quiz quiz_num.
//Post:   Each st_ave[st_num-1] contains the average for student number
//         stu_num.


void  quiz_ave   (const int grade[ ][NO_QUIZZES], double quiz_ave[ ]);
//Pre:    Each of the indexed variables grade[st_num-1, quiz_num-1]
//        contains the score for student st_num on quiz quiz_num.
//Post:   Each quiz_ave[quiz_num-1] contains the average for quiz
//        numbered quiz_num.


void  display   (const int grade[ ][NO_QUIZZES],
                    const double st_ave[], const double quiz_ave[ ]);
//Pre:    Each of the indexed variables grade[st_num-1, quiz_num-1]
//        contains the score for student st_num on quiz quiz_num. Each
//        st_ave[st_num-1] contains the average for student stu_num.
//        Each quiz_ave[quiz_num-1] contains the average for quiz
//        numbered quiz_num.
//Post: All the data in grade, st_ave, and quiz_ave have been output.
```

```
void  main( )  {
   int grade[NO_STUDENTS][NO_QUIZZES];
   double st_ave[NO_STUDENTS];
   double quiz_ave[NO_QUIZZES];
   ifstream ins;

   ins.open("in_file.dat");

   for  ( int  i = 1;  i <= NO_STUDENTS;  i++ )
        for ( int  j = 1;  j <= NO_QUIZZES;  j++ )
             ins   >>  grade [ i-1 ] [ j-1 ] ;

   st_ave(grade, st_ave);
   quiz_ave(grade, quiz_ave);
   display(grade, st_ave, quiz_ave);

   ins.close( );
   return;
}

void  st_ave  (const int grade[ ][NO_QUIZZES], double st_ave[ ])  {
   double sum;
   for (int st_num = 1; st_num <= NO_STUDENTS; st_num++) {
      sum = 0;
      for (int quiz_num = 1; quiz_num <= NO_QUIZZES; quiz_num++)
         sum = sum + grade[st_num-1][quiz_num-1];

       st_ave[st_num-1] = sum/NO_QUIZZES;
   }
}

void  quiz_ave   (const int grade[ ][NO_QUIZZES],double quiz_ave[ ]) {
   double sum;
   for (int quiz_num = 1; quiz_num <= NO_QUIZZES; quiz_num++)  {
      sum = 0;
      for (int st_num = 1; st_num <= NO_STUDENTS; st_num++)
         sum = sum + grade[st_num-1][quiz_num-1];

      quiz_ave[quiz_num-1] = sum/NO_STUDENTS;     }
}
```

```
void display (const int grade[][NO_QUIZZES],
              const double st_ave[], const double quiz_ave[])  {
   cout.setf(ios::fixed);
   cout.setf(ios::showpoint);
   cout.precision(1);

   cout << setw(10) << "Student"
        << setw(5) << "A   ve"
        << setw(15) << "Quizzes\n";
   for (int st_num = 1; st_num <= NO_STUDENTS; st_num++)  {
      cout << setw(10) << st_num
           << setw(5) << st_ave[st_num-1] << " ";

      for (int quiz_num = 1; quiz_num <= NO_QUIZZES; quiz_num++)
           cout << setw(5) << grade[st_num-1][quiz_num-1];
      cout << endl;
   }

   cout << "Quiz averages = ";
   for (int quiz_num = 1; quiz_num <= NO_QUIZZES; quiz_num++)
           cout << setw(5) << quiz_ave[quiz_num-1];
   cout << endl;
}
```

Sample output:

```
Student     Ave              Quizzes
        1  10.0        10     10     10
        2    1.0          2      0      1
        3    7.7          8      6      9
        4    7.3          8      4     10
Quiz A verage =        7.0    5.0    7.5
```

Character Strings

- sequence of characters
- stored in arrays of type *char* (ends with the null character '\0')
- difference between string and character
  - o double quotes — single quotes

    `"a" (string)` `'a' (character)`

  - o storage

    `2-element array` `byte`

  - o content

a	\0

a

- example

```
cout << "Hello world"
```

H	e	l	l	o		w	o	r	l	d	\0

```
char string1[12];
cin >> string1;     // type "Hello" as input
```

H	e	l	l	o	\0	?	?	?	?	?	?

```
char string1[12] = "Hello";
```

String I/O

- String can be output using the operator <<

  Ex:

```
char name[16] = "Chan Tai Man"
```

```
cout << endl << name;
```

C	h	a	n		T	a	i		M	a	n				

```
cout << endl << setw(16) << name;
```

				C	h	a	n		T	a	i		M	a	n

```
cout << endl << setiosflags(ios::left)
     << setw(16) << name;
```

C	h	a	n		T	a	i		M	a	n				

String I/O

- String can be input using the operator >>, but white spaces are skipped when strings are read this way .

  Ex.

  ```
  char a[80], b[80];
  cout << "Enter a string:\n" ;
  cin >> a >> b;
  cout << a << b << "END OF OUTPUT\n";
  ```

  Output:

  ```
  Enter a string:
  This is a test.
  ThisisEND OF OUTPUT
  ```

- The function **getline** can be used to read an entire line of input into a string variable.

  Ex.

  ```
  char a[80];
  cout << "Enter a string:\n" ;
  cin.getline ( a, 80 );    // The 2nd argument is the size of a
  cout << a << "END OF OUTPUT\n";
  ```

  Output:

  ```
  Enter a string:
  This is a test.
  This is a test.END OF OUTPUT
  ```

String Copy

- ***strcpy(dest, src)***
  - o copies string *src* into string *dest*
  - o example:

```
char name1[16], name2[16];

strcpy(name1,"Chan Tai Man");
```

C	h	a	n		T	a	i		M	a	n	\0	?	?	?

```
strcpy(name2,"999999999999999");
```

9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	\0

```
strcpy(name2,name1);
```

C	h	a	n		T	a	i		M	a	n	\0	9	9	\0

- ***strncpy(dest, src, n)***
  - o copies first *n* characters of *src* into *dest*

```
char name1[16];

strncpy(name1,"Chan Tai Man",16);
```

C	h	a	n		T	a	i		M	a	n	\0	\0	\0	\0

```
//Copy the value of a string to a string variable
#include <iostream.h>
#include <string.h>

void string_copy(char target[], const char source[],
                  int target_size);
//Pre:   target_size is the declared size of target.
//       source contains a string ends with '\0'.
//Post: The value of target has been set to the string
//       value in source, provided the declared size of
//       target is large enough. If target is not large
//       enough to hold the entire string, a string
//       equal to the lenght of source will be stored.

int main( )
{
    char short_string[11];
    char long_string[] = "This is rather long.";

    string_copy(short_string, "Hello", 11);
    cout << short_string << "STRING ENDS HERE.\n";

    string_copy(short_string, long_string, 11);
    cout << short_string << "STRING ENDS HERE.\n";
    return 0;
}

//Uses string.h:
void string_copy(char target[], const char source[],
                  int target_size)
{
    int new_length = strlen(source);

    if (new_length > (target_size - 1))
        new_length = target_size - 1;

    for (int index = 0; index < new_length; index++)
        target[index] = source[index];
    target[index] = '\0';
}
```

String Comparison

- ***strcmp(s1, s2)***
  - o compares strings *s1* and *s2*, returns
    - \- < 0 if *s1* < *s2*
    - \- = 0 if *s1* == *s2* *(i.e.* ***strcmp*** returns **false** *)*
    - \- > 0 if *s1* > *s2*
  - o lexicographical comparison

str1	***str2***	***return value***	***reason***
"AAAA"	"ABCD"	< 0	'A' < 'B'
"B123"	"A089"	> 0	'B' > 'A'
"127"	"409"	< 0	'1' < '4'
"abc12"	"abc12"	= 0	equal strings
"abc"	"abcde"	< 0	str1 a substring of st2
"3"	"12345"	> 0	'3' > '1'

- ***strncmp(s1, s2, n)***
  - o make at most *n* comparisons

```
strncmp("abcdefg", "abc123", 3)
returns 0
```

- ***strlen(s)***
  - o return length of string (not including '\0')

```
strlen("abcde")
returns 5
```

Some Common Errors

- Illegal to assign a value to a atring variable (except at declaration).

  Ex.
  ```
  char a_string[10];
  a_string = "Hello" ;          //  illegal assignment
  ```

  Use strcpy (a_string, "Hello") instead.

- The operator == doesn't test two strings for equality.

  Use strcmp (string1, string2) instead (but note that strcmp returns false if the two strings are the same).

- strcpy will let you assign a string that is too long for the string variable to store the value.

  Ex.
  ```
  char  short_string[3];
  strcpy ( short_string,  "This string is too long." );
  ```

  The memory locations follows the memory allocated for short_string will be filled.

```
#include <iostream.h>
#include <stdlib.h>
#include <ctype.h>

void read_and_clean(int&  n);
//Reads a line of input. Discards all symbols except the digits. Converts
//the digit string to an integer and sets n to the value of this integer              .

void new_line( );
//Discards all the input remaining on the current  line (including '\n').

int main(  )  {
   int n;
   char ans;

   do  {
      cout << "Enter an integer and press return: ";
      read_and_clean(n);

      cout << "The string converts to the integer " << n << endl;
      cout << "Again? (yes/no): ";
      cin >> ans;

      new_line( );
   }  while ( (ans != 'n') && (ans != 'N') );

   return 0;
}
```

```
//Reads a line of input. Discards all symbols except the digits. Converts
//the digit string to an integer and sets n to the value of this integer .
void read_and_clean(int& n) {
   const int ARRAY_SIZE = 6;
   char digit_string[ARRAY_SIZE];
   char next;
   int index = 0;

   cin.get(next);
   while (next != '\n') {
      if ( (isdigit(next)) && (index < ARRAY_SIZE - 1) ) {
            digit_string[index] = next;
            index++;
      }
      cin.get(next);
   }
   digit_string[index] = '\0';

   n = atoi (digit_string);     // the function atoi convert a string to integer
}

//Discards all the input remaining on the current line (including '\n').
void new_line( ) {
   char symbol;
   do {
      cin.get(symbol);
   } while (symbol != '\n');
}
```

Sample output:

Enter an integer and press return: **$ 100**
The string converts to the integer 100
Again? (yes/no): **yes**
Enter an integer and press return: **99%**
The string converts to the integer 99
Again? (yes/no): **yes**
Enter an integer and press return: **23% &*5 * 13**
The string converts to the integer 23513
Again? (yes/no): **no**