Arrays, Strings, and Files

1. Arrays
   - memory allocation
   - multi-dimensional arrays
2. Strings
   - strings are arrays
   - use of getline
3. Files
   - file streams
   - copying text files line by line

MCS 360 Lecture 3
Introduction to Data Structures
Jan Verschelde, 17 January 2020

Arrays, Strings, and Files

1. Arrays
   - memory allocation
   - multi-dimensional arrays
2. Strings
   - strings are arrays
   - use of getline
3. Files
   - file streams
   - copying text files line by line

Memory Allocation

When we declare variables, memory is allocated:

```
double x;
int a[5];
```

The variable `x` occupies space for one `double`
and the array `a` is a sequence of 5 `int`s.

Instead of allocating memory at declaration, we can

- declare a pointer to a variable;
- use `new` to allocate memory.

```
int *a;
// intermediate statements
a = new int[5];
```

Allocation

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

int main()
{
   int n;

   cout << "give the dimension : ";
   cin >> n;

   cout << "allocating an array of length "
        << n << endl;

   double a[n];
   for(int i=0; i<n; i++) a[i] = double(i);
```

dynamic allocation and deallocation

```
    double *b;
    b = new double[n]; // dynamic allocation
    for(int i=0; i<n; i++) b[i] = a[i];

    for(int i=0; i<n; i++)
       cout << "  a[" << i << "] = " << a[i];
    cout << endl;
    for(int i=0; i<n; i++)
       cout << "  b[" << i << "] = " << b[i];
    cout << endl;

    delete[] b; // deallocation: release memory

    return 0;
}
```

Arrays, Strings, and Files

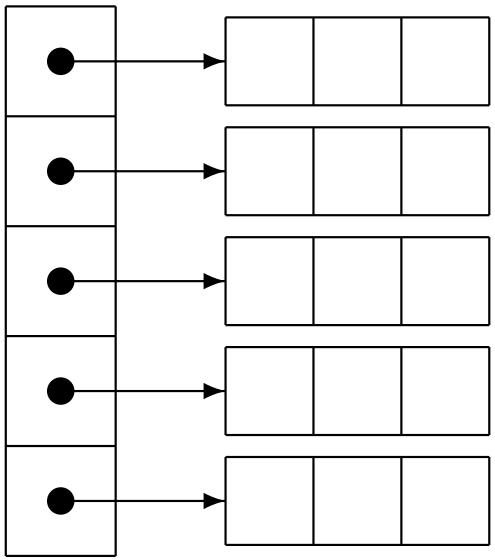
1. Arrays
   - memory allocation
   - multi-dimensional arrays
2. Strings
   - strings are arrays
   - use of getline
3. Files
   - file streams
   - copying text files line by line

multi-dimensional arrays

Declaring a 5-by-3 matrix `A` of doubles:

```
double A[5][3];
```

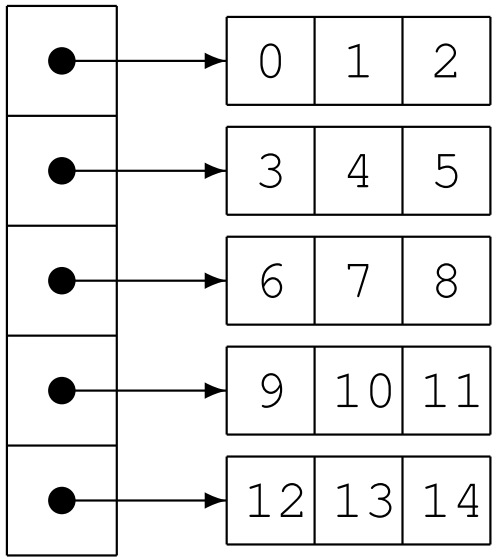
The above declaration allocates space for a matrix with 5 rows and 3 columns:



double loops

Working with double arrays leads to double loops:

```
for(int i=0; i<5; i++)
   for(int j=0; j<3; j++)
      A[i][j] = double(i+j);
```



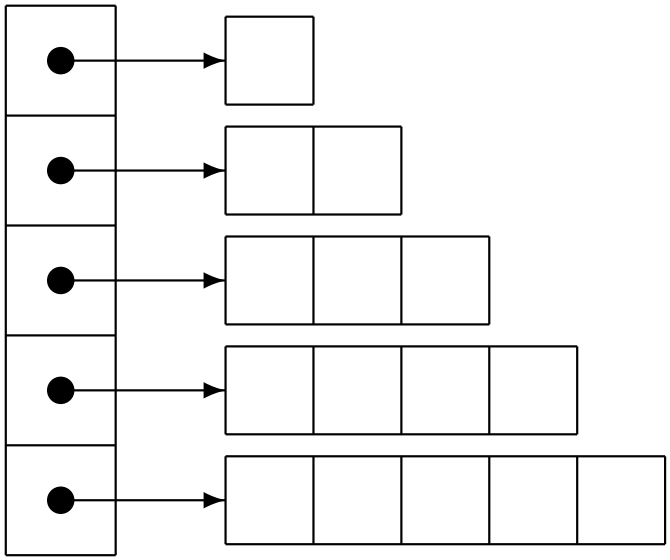
Beware of the braces when printing:

```
for(int i=0; i<5; i++)
{
   for(int j=0; j<3; j++)
      cout << " " << A[i][j];
   cout << endl;
}
```

a triangular matrix

```
int* a[5];
for(int i=0; i<5; i++)
    a[i] = new int[i+1];
```

The effect of the above statement is a triangular matrix:



Arrays, Strings, and Files

1. Arrays
   - memory allocation
   - multi-dimensional arrays
2. Strings
   - strings are arrays
   - use of getline
3. Files
   - file streams
   - copying text files line by line

Strings are Arrays

Strings are arrays of characters, e.g.:

```
char line[80];
```

C++ provides a class `string`.

Stretching a given string:

```
$ strings_are_arrays
give a string : hello
received 5 characters
stretched string : h e l l o
```

The new string is a `C` string.

the `length()` method

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
   string s;

   cout << "give a string : ";
   cin >> s;

   int n = s.length();

   cout << "received " << n
        << " characters" << endl;
```

stretching the string

```
    char ss[2*n];

    ss[0] = s[0];
    for(int i=1; i<n; i++)
    {
       ss[2*i-1] = ' ';
       ss[2*i] = s[i];
    }
    ss[2*n-1] = '\0'; // end of string symbol

    cout << "stretched string : "
         << ss << endl;

    return 0;
}
```

Arrays, Strings, and Files

1. Arrays
   - memory allocation
   - multi-dimensional arrays
2. Strings
   - strings are arrays
   - use of getline
3. Files
   - file streams
   - copying text files line by line

reading strings

With the extraction operator >> we read till the end of line or the first space.

Extending our `hello_there` from lecture 1:

```
$ use_getline
Enter your full name : John Smith
Hello Dr. Smith, may I call you John?
That was fun, let us try it again!
Enter your full name : John Smith
you entered John Smith
Hello Dr. Smith, may I call you John?
```

The user types twice a full name, separated by a space.

console input

```
#include <iostream>
#include <string>
#include <limits>

using namespace std;

int main()
{
   string given_name, family_name;

   cout << "Enter your full name : ";

   cin >> given_name >> family_name;

   cout << "Hello Dr. " << family_name
        << ", may I call you " << given_name
        << "?" << endl;
```

skipping newline

```
cout << "That was fun, let us try it again!"
     << endl;

string full_name;

cout << "Enter your full name : ";

// we must first skip the unprocessed \n symbol
// ignoring unlimited number of characters

cin.ignore(numeric_limits<int>::max(),'\n');

getline(cin,full_name,'\n');  // read till \n
```

splitting a string

```
   cout << "you entered " << full_name << endl;

   {
      size_t L = full_name.length();
      size_t p = full_name.find(" ");
      string first = full_name.substr(0,p-1);
      string second = full_name.substr(p+1,L-1);

      cout << "Hello Dr. " << family_name
           << ", may I call you " << given_name
           << "?" << endl;
   }

   return 0;
}
```

Arrays, Strings, and Files

1. Arrays
   - memory allocation
   - multi-dimensional arrays
2. Strings
   - strings are arrays
   - use of getline
3. Files
   - file streams
   - copying text files line by line

file streams

Files store data permanently.
A common text file format is `csv`,
`csv` = comma separated values.

Once opened for input or output,
files are used like console input or output:

- same syntax for insertion `<<`
  and extraction operators `>>`
- text files are often processed line by line,
  using the `getline` function.

Reading from and writing to file is buffered,
use the method `flush()` to empty buffer to file.
Also closing a file with `close()` flushes the buffers.

Arrays, Strings, and Files

1. Arrays
   - memory allocation
   - multi-dimensional arrays
2. Strings
   - strings are arrays
   - use of getline
3. Files
   - file streams
   - copying text files line by line

input and output streams

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

int main()
{
   string input_file_name, output_file_name;

   cout << "Give name of input file : ";
   cin >> input_file_name;
   cout << "Give name of output file : ";
   cin >> output_file_name;

   ifstream ins(input_file_name.c_str());
   ofstream outs(output_file_name.c_str());
```

Names of files are C strings!

copying line by line

```
    string buffer;
    int count = 0;

    while(!ins.eof()) // not at end of file
    {
       getline(ins,buffer,'\n');
       outs << buffer;
       outs.flush();   // optional flushing
       if(!ins.eof()) outs << '\n';
       count = count + 1;
    }
    cout << "... copied " << count
         << " lines" << endl;

    ins.close(); outs.close();

    return 0;
}
```

Summary + Exercises

We ended a first tour of Chapter P.
Important to know how to get data:

1. input via console input (keyboard);
2. generate at random via `rand()`;
3. read from file.

Exercises: Write C++ programs for the tasks below:

1. Revert the characters in a given string, e.g.: if we type `hello`, the program replies with `olleh`.
2. Ask the user for the number of rows and columns of a matrix. Generate a matrix of random integer numbers in the range 0..1000 (see lecture 2). Find the location (row and column index) of the largest element in the matrix.
3. Compute the number of semicolons of a C++ program on file. Let the user enter the name of the file.

more exercises

4. A plateau in an array is the longest sequence of the same elements that occur in the array.
   Define a function that returns the start and end index of a plateau in a given array of integers.
5. Write a function which takes as its input argument a string. The string represents a line in comma separated value format. The values between the commas are integers. The function returns an array with the integers in the comma separated value string. You may assume that the input string is correct and contains at least one integer number.
6. A saddle point in a matrix A is maximal in its row; and minimal in its column. Write the definition of a function which takes on input an n-by-m matrix A with integer values and returns the value of the saddle point in A.