RWTH Aachen University
Center for Computing and Communication, MATSE-Training

B. Küppers, M. Politze, A. Voß

Recap C/C++ for parallel programming 27.02.2012

Exercise 1

Read in a variable of type `int`. This works as follows:

```
cout << "Please enter a number: ";
cin >> n;
cout << "The number is: " << n << endl;
```

a) What happens when you do not enter a number?

b) Read a string (`char*` or `string`) variable and print it. What is the difference?

Exercise 2

a) The factorial is computed recursively as follows. Copy and paste the given function *above* the `main` function and call it from the `main` function.

```
int factorial(int n)
{
      if (n <= 1)
            return 1;
      else
            return n*factorial(n-1);
}
```

Read `n` form user input, calculate the faculty and print it.

Copy and paste the function below the `main` function. What happens? Insert the following declaration above `main` and try it again:

```
int faculty(int n);
```

b) Declare and define a recursive function `fib` that calculates according to the following recursive formula

```
fib(n) = fib(n-1) + fib(n-2) for n>=2,
fib(1) = 1,
fib(0) = 0
```

the n^{th} Fibonacci number. Again read `n` from the console and print the n^{th} Fibonacci number.

Exercise 3

Move the functions in a different file (e.g. mymath.h) and import them using the `#include` directive.

Exercise 4

Rewrite the following code from Java to C++. Consider the following:

- `static` does not have to be taken care of.
- Read `LETZTEZEILE` from console.

Split your program into the same functions.

```
public class Pascaldreieck {
  final static int LETZTEZEILE = 4;   // => 5 Zeilen

  public static void main(String[] kommandozeile) {
    for (int n = 0; n <= LETZTEZEILE; ++n)
      dreieckszeileAusgeben(n, LETZTEZEILE);
  }

  static void dreieckszeileAusgeben(int n, int m) {
    for (int k = m - n; k >= 1; --k)  // m-n Leerzeichen
      System.out.print("␣");          // ausgeben
    for (int k = 0; k <= n; ++k)
      System.out.print("␣" + binomialkoeffizient(n, k));
    System.out.println();
  }

  static int binomialkoeffizient(int n, int k) {
    int ergebnis = 1;
    for (int i = 1; i <= k; ++i)
      ergebnis = (ergebnis * (n - i + 1)) / i;
    return ergebnis;
  }
}
```

Exercise 5

The following code defines „constants“ and „functions“, called macros using the precompiler.

```
// Constant
#define PI 3.14156
// "Function"
#define max(n,m) ((n>m) ? n : m)

// in main

#if defined(PI)
      cout << "Pi = " << PI << endl;
#else
      cout << "Pi unbekannt " << endl;
#endif
      cout << "max(3,4) = " << max(3,4) << endl;
```

Take the given code snippets, compile them and try to understand what happens.

a) Now call the `max` macro like shown. What values do `m` and `n` have afterwards?

```
int n=3, m=4;
cout << "max(n,m) = " << max(++n,++m) << endl;
cout << "n=" << n << ", m=" << m << endl;
```

b) Now define a function for multiplication that can be called as the following example shows. What happens and why?

```
#define AmulB(a,b) (a * b)

int a=2, b=3, c=4;
cout << "A * B: " << AmulB(a,b+c) << endl;
```

Can you solve the problem by changing the macro?

Exercise 6

Declare:

- an integer variable `n` and a constant integer `m`,
- a pointer `p1` to an integer variable, a pointer `p2` to a constant integer variable, a constant pointer `p3` to an integer variable and a constant pointer `p4` to an constant integer.

```
int*             p1 = &n;
const int*       p2 = &n;
int* const       p3 = &n;
const int* const p4 = &n;
```

Initialize all pointers with the address of `n`.

Try to make the following changes and try to compile / execute your program:

- Change the value of the variable pointed to.
- Change the address pointed to, to the address of `m`.

Discuss why certain assignments are not possible.

Exercise 7

Implement two variants of a function to swap three variables:

```
int a = 12, b = 16, c = 32;
Swap(&a,&b,&c);  // a=16 b=32 c=12
Swap(a,b,c);     // a=32 b=12 c=16
```

Implement a variant using pointers and a variant using references.

Exercise 8

The given code reserves memory (size given in bytes) using `malloc` and initializes it with values according to the position in the buffer.

```
const int n = 16;
void* p = (char*)malloc(n*sizeof(char));
```

The following three loops are equivalent. Why?

```
const int lenChar = n/sizeof(char);
char* pc = (char*)p;

for (int i=0; i<lenChar; ++i)
      pc[i] = 65+i;
for (int i=0; i<lenChar; ++i)
      *(pc+i) = 65+i;
for (char* pp = pc; pp<pc+lenChar; ++pp)
      *pp = 65+(pp-pc);
```

When the programm is finished the memory is freed using free.

```
free(p);
```

Implement three equivalent loops that fill the buffers with integers instead of characters. How many integers can you fit in the buffer?

Exercise 9

In C strings are an array of characters terminated with the 0-character (`(char)0`).
Write a function that has the following signature:

```
int PosZero(const char * pc)
```

that uses pointer arithmetic to determine the length of a string. Do not use integer variables to solve this problem. Try to call the function like follows:

```
int pos;
pos = PosZero("Hallo");        // 5
pos = PosZero("");             // 0
pos = PosZero("1234567890");   // 10
```

Write a function

```
char* TextCopy(const char* pc)
```

that allocates enough memory to store the given string (`malloc`) and copy the given string to the new buffer. Return a pointer to the copy. Do not forget the 0-character at the end. The function shall be called as follows:

```
char* pc = TextCopy("Hallo");
cout << "Txt '" << pc << "'" << endl;
free(pc);
```

Exercise 10

Write a function as follows:

```
void Sieve(int MaxPrim, int& SizeOfA, int*& A)
```

The function uses the Sieve of Eratosthenes to calculate prime numbers up to `MaxPrim`. These values are saved and returned in the array `A` with the size `SizeOfA`. The array `A` will be allocated in the function according the number of prime numbers found. Thus the memory has to be freed by the user of the function `Sieve`.

Use `malloc` and `free` to allocate and free the memory used for storing the prime numbers.

Print the prime numbers found to the console.

Exercise 11

Write and test a function that reverses a string.

```
void Reverse(const char* sSrc, char*& sDst)
```

Exercise 12

Write and test a function that will save the 32 bits of an integer number in a character array. Example:

```
char buffer[4*(8+1)]; // xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxxxx/0
unsigned int n;
n=0x12345678; ConvertIntToBitString(n,buffer);
cout << "n=" << n << ", '" << buffer << "'" << endl;
```

with output:

```
n=305419896, '00010010 00110100 01010110 01111000'
```

The function has the following signature:

```
void ConvertIntToBitString(unsigned int n, char buffer[])
```

Exercise 13

The file `testData.txt` contains data measured by some physical experiment:

```
11
0 0
0.628319 0.587785
1.25664 0.951057
1.88496 0.951057
2.51327 0.587785
3.14159 0
3.76991 -0.587785
4.39823 -0.951057
5.02655 -0.951057
5.65487 -0.587785
6.28319 0
```

The first line gives the number of experiments. The following lines contain pairs of values from xy-space.

Every x-y-tupel is saved in a `struct` called `Point`. The `struct` has only two properties, x and `y`.

a) Write a function

```
void readData(const char* pcFilename, unsigned int& szData, Data*& data)
```

that reads the data from the file and stores it in an array of `Point`. You can allocate this array by using `(Data)malloc(n*sizeof(Data))`. The size of the array is returned using the `szData` parameter.

- Read the file by using streams (`ifstream`).
- Tip: Use the function `strchr` to find a particular character in a string.
- Tip: To convert a string into an integer or double use atoi or atof respectively.
- Tip: Use the reference http://en.cppreference.com/w/cpp/string/narrow to find other string functions.

b) To analyze the given data statistical measures like mean and standard deviation can be used. Write a method:

```
void analyzeData(unsigned int szData, const Data* data,
                 double& m, double& s)
```

that calculates the mean m and the standard deviation s. You can find the definition of mean and standard deviation by searching the web.

c) Consider the data to be a function that only depends on the values of the given sampling points. Write a function that can interpolate between the given samples:

```
double evalData(const unsigned int szData, const Data* data, double x)
```

Also consider the following:

- If `x` is smaller than the smallest x-value given in the data the function will return the y-value of the smallest x-value. A bigger `x` than the biggest x-value is handled respectively
- If `x` is between two points linearly interpolate between the two points to calculate the according y-value.