

Programming Exercise 8 (MATH)

Due Date and Time: June 30, 2009, 23:59

You should work out the exercises independently. Group cooperation is not allowed. Submit the sensibly documented program files. (Files for Bsp. 5: `bsp_5.cpp`, `bsp_5_fkt.cpp`, `bsp_5_fkt.hpp`) Copy these to your directory in the global submission folder.

29. Replace the static C fields in problem 19 (Exercise 6) with the vector class `vector<int>` for which allows a dynamic length of the vector at run time. See for more information §2.3.4¹ of the lecture notes.

Rewrite your functions from problem 19 for the above vector class and save them separately in header and source code files (3 files are to be submitted). Use the method of `size` der vector classes. (1 Pt.)

Explore the functionality of the methods `push_back`² and `pop_back` of the vector class. Test these methods out.

Eingabedaten (n): (4), (20), (31553)

30. Write a class , which stores Day, Month, and Year. Implement (Header and source files in `bsp_30_fkt.hpp` and `bsp_30_fkt.cpp`), such that the following program³ works. (3 Pts.)

```
1  #include <iostream>
2  #include "bsp_30_fkt.hpp"
3  using namespace std;
4  \
5  int main()
6  {
7      Datum aa( 4, 12, 1983);
8      Datum bb(aa);
9      Datum cc(31,  5, 1983);
10     Datum dd;
11     \
12     dd = cc;
13     cout << "dd : " << dd << endl;
14     if ( aa < dd ) // aa is an earlier date than dd?
15         { cout << aa << " earlier than " << dd << endl; }
16     else
17         { cout << aa << " not earlier than" << dd << endl; }
18     \
19     if ( bb == aa )
20         { cout << "aa and bb are equal" << endl; }
21     \
22     return 0;
23 }
```

¹<http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/Script/html/node26.html>

²http://www.cplusplus.com/reference/stl/vector/push_back

³http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/SS09/uebung8/bsp_30.cpp

Hint: Implement either first the <<-operator for your class DATUM or comment out the corresponding rows in the beginning of the program. Otherwise a lengthy, cryptic, error message will appear at compile time.

31. Implement a class for the coefficients of a polynomial of degree n . This class should contain a Standard- copy constructor, destructor, and a parameter constructor with a file name. The latter should read in the data from a file, see (§ 9.2 of the lecture notes⁴). Furthermore the allocation and return operators (4 Pts.) should be implemented. The evaluation of the polynomial

$$p_n(x) = \sum_{k=0}^n a_k \cdot x^k$$

on a point x should be realized as a method (member function) of the class. Evaluate the polynomial in a second method using the HORNER scheme.

Save the class declaration and implementation in the files *bsp_31_fkt.hpp* und *bsp_31_fkt.cpp*.

Testdaten (Filename, x): (data_31_a.txt⁵, 3.78), (data_31_b.txt⁶, -1.0087), (data_31_c.txt⁷, -0.945),

The ASCII-(Text-)Files contain n and a_k , $k = 0, \dots, n$.

32. Further develop your class from problem 31, so that (2 Pts.)

- the class member is only visible within the member functions,
- the class member is can be read using (Inline-) functions,
- all methods which can not modify class members are indicated as such.

Write a short test program for this

33. Implement a class Sparschwein (piggy bank), which has the following data elements:

- Counter for the four kinds of coins (1-Cent, 10-Cent, 50-Cent, 1-Euro)
- The maximum number of coins that fit in the piggy bank.
- A flag, which indicates if the piggy bank is broken open.

(3 Pts.)

The class should contain the following elements

- `init(...)` initializes an empty piggy bank,
- `add1Cents(...)` puts in a 1-Cent-coin,
- `add10Cents(...)`, `add50Cents(...)`, `add1Euro(...)` as above,

⁴<http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/Script/html/node91.html>

⁵http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/SS09/uebung8/data_31_a.txt

⁶http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/SS09/uebung8/data_31_b.txt

⁷http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/SS09/uebung8/data_31_c.txt

- `isEmpty(...)` Checks if the piggy bank is empty,
- `isFull(...)` Checks if the piggy bank is full,
- `isBroken(...)` Checks if the piggy bank is broken open,
- `breakInto(...)` Breaks the piggy bank open, and returns the saved money.

Use header and source files for the implementation of the class. The main program (an additional source file) should put coins in the piggy bank until it is full and then crack it open. Show the collected quantity. After this put 37 Cents in the piggy bank and crack it open again. (How much money is in there?)

34. Use the classes `string` and `vector` from the STL for this task. Look at §2.3.1 and §2.3.4 of the lecture notes (respectively. `string`⁸ and `vector`⁹). You can also look at an example for `vector` in §3.3.1 of the Download–Book of Breymann¹⁰ and `string` in the lecture Schröder¹¹. (4 Pts.)

Declare a class for the performance of a Handy (mobile phone), so that the device name, owner name, telephone name, and monthly fees are characterized. This class should contain (if needed) the usual constructors access methods of the member and the return the complete data of the phone.

In addition to the usual constructors (in case they are needed), this class should provide access methods for the member data and which can return the complete data for the mobile phone.

Use this Handy class to make an address for your friends (you must have more than 5). How would it be with an address book class?

Read in the unsorted data from a file and use the and use the function `getline`¹² to read in entire lines, as well as empty lines.

Return your sorted address book. (Tip: in the above given web references look for an algorithm `sort`).

Save your class declaration and implementation in the files `bsp_34_fkt.hpp` und `bsp_34_fkt.cpp`.

Turn in your data file(s) `bsp_34_data.txt`.

General guidelines:

The commands `break`, `continue`, `goto` are **not allowed**.

⁸<http://www.cplusplus.com/reference/string/>

⁹<http://www.cplusplus.com/reference/stl/vector/>

¹⁰<http://www.informatik.hs-bremen.de/~brey/stlb.html>

¹¹http://www.cpp-tutor.de/cpp/1e06/1e06_02.htm

¹²http://www.cpp-tutor.de/cpp/1e06/1e06_02.htm

Submission of solutions:

The delivery of solutions (*.cpp-Files, *.hpp, ..., **no exe-Files**) is exclusively through the copying of your files in your (on first time created) directory `Lastname_Firstname` in the global submission folder `y: \\pers.ad.uni-graz.at\fs\ou\621\stud_haase\Nachname_Vorname .`

Guidelines beyond this can be found at: LV-Homepage¹³

The file names **must** follow the scheme `bsp_nummer` with the corresponding file extension. For example, in example 1 the file `bsp_1.cpp` would be submitted. Other file designations will not be accepted.

¹³<http://www.uni-graz.at/~haasegu/Lectures/Kurs-C/SS09/index.html>