OBRADA IZUZETAKA

1. Projektovati klasu za obradu vektora realnih brojeva sa zadatim opsezima indeksa. Za razrešavanje konfliktnih situacija koristiti mehanizam obrade izuzetaka. Sastaviti glavni program za prikazivanje mogućnosti te klase.

```cpp
#pragma once
#include <iostream>
#include <string>
using namespace std;

class Vektor
{
   int max_ops, min_ops;
   double *niz;
public:
   enum Greska {
      OK,
      OPSEG, // neispravan opseg indeksiranja
      MEMORIJA, //dodela memorije nije uspela
      PRAZAN, //vektor je prazan
      INDEKS, //indeks je izvan opsega
      DUZINA //neusaglasene duzine vektora
   };
   Vektor() { niz = 0; }
   Vektor(int, int = 0);
   Vektor(const Vektor&);
   ~Vektor();
   Vektor& operator=(const Vektor&);
   double& operator[](int) const;
   friend double operator*(const Vektor&, const Vektor&); // skalarni proizvod
};
```

Vektor/Vektor.h

```cpp
#include "Vektor.h"

Vektor::Vektor(int a, int b) : max_ops(a), min_ops(b)
{
   if (max_ops < min_ops) throw(Greska::OPSEG);
   else if (!(niz = new double[max_ops - min_ops + 1]))
      throw(Greska::MEMORIJA);
   else for (int i = 0; i < max_ops - min_ops + 1; i++)
   {
      niz[i] = 0;
   }
}

```

```
14 Vektor::Vektor(const Vektor& a) : max_ops(a.max_ops), min_ops(a.min_ops)
15 {
16    if (!(niz = new double[max_ops - min_ops + 1]))
17       throw (Greska::MEMORIJA);
18    for (int i = 0; i < max_ops - min_ops + 1; i++)
19    {
20       niz[i] = a.niz[i];
21    }
22 }
23
24 Vektor::~Vektor()
25 {
26    delete[] niz;
27    niz = 0;
28 }
29
30 Vektor& Vektor::operator=(const Vektor& a)
31 {
32    if (&a != this)
33    {
34       delete[] niz;
35       niz = 0;
36       max_ops = a.max_ops;
37       min_ops = a.min_ops;
38       if (!(niz = new double[max_ops - min_ops + 1])) throw (Greska::MEMORIJA);
39       else for (int i = 0; i < max_ops - min_ops + 1; i++)
40       {
41          niz[i] = a.niz[i];
42       }
43    }
44    return *this;
45 }
46
47 double& Vektor::operator[](int i) const
48 {
49    if (!niz) throw(Greska::PRAZAN);
50    else if ((i < min_ops) || (i > max_ops))
51       throw(Greska::INDEKS);
52    else return niz[i - min_ops];
53 }
54
55 double operator*(const Vektor& a1, const Vektor& a2)
56 {
57    if ((!a1.niz) || (!a2.niz))
58       throw(Vektor::PRAZAN);
59    else if (a1.max_ops - a1.min_ops != a2.max_ops - a2.min_ops)
60       throw(Vektor::DUZINA);
61    else
62    {
63       double s = 0;
64       for (int i = 0; i < a1.max_ops - a1.min_ops + 1; i++)
65       {
66          s += a1.niz[i] * a2.niz[i];
67       }
68       return s;
69    }
70 }
```

Vektor/Vektor.cpp

```cpp
#include "Vektor.h"

int main()
{
   while (1)
   {
      try
      {
         int max, min;
         cout << "Unesi opseg indeksa prvog vektora: " << endl;
         cin >> max >> min;
         Vektor v1(max, min);
         cout << "Komponente prvog vektora?" << endl;
         for (int i = min; i <= max; i++)
         {
            cin >> v1[i];
         }
         cout << "Unesi opseg indeksa drugog vektora:" << endl;
         cin >> max >> min;
         Vektor v2(max, min);
         cout << "Komponente drugog vektora?" << endl;
         for (int i = min; i <= max; i++)
         {
            cin >> v2[i];
         }
         cout << "Skalarna proizvod: " << v1 * v2 << endl;
      }
      catch (Vektor::Greska g)
      {
         char *poruke[] =
         {
            "",
            "Neispravan opseg indeksa!",
            "Neuspelo dodeljivanje memorije!",
            "Vektor je prazan!",
            "Indeks je van opsega!",
            "Duzine vektora nisu iste!"
         };
         cout << poruke[g] << endl;
      }
   }
   return 0;
}
```

Vektor/Source.cpp

2. Implementirati na programskom jeziku C++ klase za rad sa fajlovima. Fajl ima logičku putanju na medijumu za čuvanje podataka, može da se obriše, da se proveri da li fajl postoji. Pisac (Writer) može da upiše tekstualni sadržaj u pridruženi fajl, može mu se pridružiti fajl za pisanje i može se završiti pisanje. Ukolio se pokuša pridruživanje novog fajla, iako stari jozv s nije zatvoren, to je greška. Greška je i ako fajl ne može da se pronađe ili obriše. Ukoliko pisac ne može da upiše sadržaj u fajl, treba da prijavi grešku.

```cpp
#include <exception>
#include <iostream>
#include <fstream>
#include <string>
```

```
5  using namespace std;
6
7  class FileException;
8  class FileNotFoundException;
9  class FileNotDeletedException;
10 class FileAlreadyAssignedException;
11 class FileAccessException;
12 class FileNotClosedException;
13
14 class File
15 {
16    string filePath;
17 public:
18    File(const string& path) : filePath(path) { }
19    void deleteFile();
20    bool exists() const;
21    const string& getPath() const { return filePath; }
22 };
23
24 class Writer
25 {
26    File* assignedFile;
27    ofstream* assignedStream;
28 public:
29    Writer() : assignedFile(NULL), assignedStream(NULL) {}
30    void assignFile(const File& file);
31    void close();
32    void write(const string& text);
33    const File* getAssignedFile() const { return assignedFile; }
34 };
35
36 class FileException : public exception
37 {
38    File* wrongFile;
39 public:
40    FileException(const string& msg, const File* file = NULL);
41    virtual ~FileException() { delete wrongFile; }
42    File* getWrongFile() const { return wrongFile; }
43 };
44
45 class FileNotFoundException : public FileException
46 {
47 public:
48    FileNotFoundException(const string& msg, File* file = NULL)
49       : FileException(msg, file) {}
50 };
51
52 class FileAccessException : public FileException
53 {
54 public:
55       FileAccessException(const string& msg, File* file = NULL) : FileException(msg↩
              , file) {}
56 };
57
58 class FileAlreadyAssignedException : public FileAccessException
59 {
60 public:
61    FileAlreadyAssignedException(const string& msg, File* file = NULL)
62       :FileAccessException(msg, file) {}
63 };
```

```
64
65 class FileNotDeletedException : public FileAccessException
66 {
67 public:
68    FileNotDeletedException(const string& msg, File* file = NULL)
69       :FileAccessException(msg, file) {}
70 };
71
72 class FileNotClosedException : public FileAccessException
73 {
74 public:
75    FileNotClosedException(const string& msg, File* file = NULL)
76       :FileAccessException(msg, file) {}
77 };
```

Fajlovi/File.h

```
1 #include "File.h"
2
3 bool File::exists() const
4 {
5    ifstream infile(filePath);
6    return infile.good();
7 }
8
9 void File::deleteFile()
10 {
11    if (!exists()) throw FileNotFoundException(string("File not found: ") + filePath↩
       , this);
12    if (remove(filePath.c_str()) != 0)
13       throw FileNotDeletedException(string("File not deleted: ") + filePath, this);
14 }
15
16 void Writer::close()
17 {
18    assignedFile = NULL;
19    if (assignedStream)
20       assignedStream->close();
21    assignedStream = NULL;
22 }
23
24 void Writer::write(const string& text)
25 {
26    char* buffer = NULL;
27    try {
28       buffer = new char[text.size() + 1]; // dinamicki alocirana memorija
29       for (int i = 0; i < text.size(); i++)
30       {
31          buffer[i] = toupper(text.at(i));
32       }
33       buffer[text.size()] = '\0';
34       if (assignedStream == NULL || assignedStream->bad()) {
35          string msg = (assignedFile != NULL) ? assignedFile->getPath() : "File not ↩
              assigned";
36          throw FileAccessException(string("Could not write to file: ") + msg, ↩
              assignedFile);
37       }
38       *assignedStream << buffer << endl;
39       delete[] buffer; // na kraju se brise bafer, ali...
```

```
40                    // sta ako se pre toga pojavi izuzetak? Ova linija bice ↩
                          preskocena.
41       buffer = NULL;
42    }
43    catch (FileAccessException& e) { // Zato, hvatamo izuzetak samo da bismo ↩
          oslobodili
44                             // zauzete resurse.
45                             // Oslobadjaju se resursi
46       if (buffer != NULL) delete[] buffer;
47       throw e; // Prosledjuje se izuzetak dalje.
48    }
49 }
50
51 void Writer::assignFile(const File& file)
52 {
53    if (assignedFile != NULL)
54       throw FileAlreadyAssignedException(string("There is already assigned file: ")↩
              + file.getPath(), &File(file));
55    if (!file.exists())
56       throw FileNotFoundException(
57          string("Could assign file because, file not found:") + file.getPath(),
58          &File(file));
59    assignedFile = new File(file);
60    assignedStream = new ofstream(assignedFile->getPath().c_str(), ios::app);
61 }
62
63 FileException::FileException(const string& msg, const File* file) : exception(msg.↩
       c_str())
64 {
65    if (file != NULL)
66       wrongFile = new File(*file);
67    else wrongFile = NULL;
68 }
```

Fajlovi/File.cpp

```
1  #include "File.h"
2
3  void safeDelete(File* f)
4  { // Sluzi za brisanje fajlova u granama za obradu izuzetaka,
5    // pa ne sme da baci nijedan izuzetak.
6     try {
7        if (f) f->deleteFile();
8     }
9     catch (FileException& e) {
10       cout << "Could not safely delete file: " << e.what() << endl;
11    }
12 }
13
14 void safeClose(Writer& w)
15 { // Sluzi za zatvaranje u granama za obradu izuzetaka,
16   // pa ne sme da baci nijedan izuzetak.
17    try {
18       w.close();
19    }
20    catch (FileNotClosedException& e) {
21       cout << "Could not safely close file: " << w.getAssignedFile() << endl;
22    }
23 }
```

```
24
25 int main()
26 {
27    Writer w;
28    File f(string("e:\\text.txt"));
29    try {
30       w.assignFile(f);
31       w.write(string("This is some text!"));
32       // w.assignFile(f); // Ova linija koda sluzi za testiranje izuzetka.
33       w.close();
34    }
35    catch (FileNotFoundException& e) {
36       cout << "ERROR: " << e.what() << endl;
37    }
38    catch (FileAccessException& e) { // Na pocetku specijalniji izuzeci...
39       cout << "ERROR: " << e.what() << endl;
40       safeClose(w);
41    }
42    catch (FileException& e) { // Najopstiji tip izuzetka na kraju..
43       cout << "ERROR: " << e.what() << endl;
44       safeClose(w);
45       safeDelete(e.getWrongFile());
46    }
47    return 0;
48 }
```

Fajlovi/Source.cpp