Тема 3 | Работа с fstream - особености при отваряне на файл за четене и писане. Двоични файлове. Запазване на обекти в двоичен файл. Четене на обекти от двоичен файл.

fstream
- позволява писане и четене едновременно
- има само един указател

```
tellp() == tellg();
```

⚠ потоците работят с вътрешен буфер, който при достигане на определен капацитет записва.

Когато изпълним put() трябва да синхронизираме, за да направим състоянието правилно put и get

Пример:
```
fstream fs("file.txt", std::ios::in|std::ios::out);
{
  fs.put("a");
  fs.flush();
  fs.seekg(0, std::ios::begin);
  fs.get();
```

Двоични файлове
- поредица от байтове
- използва само него
- по-малък от нормален файл
- по-бърз при четене от него

⚠ ios::binary
- отваря файл в бинарен формат
- чете байт по байт

\n - чете '\' и 'n' с което маха спец. символ

Пример: file.txt

721

стринг => число -> стринг

file.dat

A7	B2	00	00

-> число

Запазване на обекти в двоични файлове

- ключова ф-ция .write(<указател>, <дължина>);

const char* ук.
избран е защото е 1 байт и const, защото дали да бъде променян

брой byte-ове, които трябва да бъдат запазени

Пример:

```
int x=16;
.write((const char*) &x, sizeof(int));
```

⚠ В двоични файлове пише в edianness-а на системата

file.dat

01	00	00	00

<- hex view of the file

Пример: писане на
- статичен обект
- обект с външен ресурс
- масив от обекти без дин. ресурс
- масив от обекти с динамичен ресурс

Четене на обекти от двоичен файл

- ф-цията .read(<указател>, <дължина>);

$\downarrow$ char* без const, защото ще пишем в тези данни.

$\downarrow$ размер, който искам да прочета

Пример:
```
ifstream ifs("file.dat", ios::binary | ios::in);
    int z;
    ifs.read((char*)&z, sizeof(z));
    z == 16;
```

Пример:

Четене на
- статичен обект
- обект с външен ресурс
- масив от обекти без дин. ресурс
- масив от обекти с динамичен ресурс

ReplaceSymbolWith
Another.cpp

```
include <iostream>
#incqudea <fstream>
void SwitchSymbol(std::fstream& fs, char ch1, char ch2)
{
  whiee (!fs.eof())
  {
     char curr=fs.get();
     if(curr==ch1)
     {
        fs.seekg(-1, std::ios::cur);
        fs.put(ch2);
        fs.flush();
     }
  }
}

void SwitchSymbol(const char*fileName, char ch1, char
                                                 ch2)
{
  std::fstream fs(fileName, std::ios::in | std::ios::out);

   assert(fs.is_open());

  SwitchSymbol(fs, ch1, ch2);
  fs.close();
```

(ArrayStudentsStaticName.cpp)

```
#include <iostream>
#include <fstream>
using namespace std;
Struct Student
{
   char name [30];
    int  age;
    int  fn;
};
Void initStudent(Student& st, const char* name, int age,
                                              int fn)
 { strcpy (st.name, name);
   st.age = age;
   st.fn = fn;
 }
oid saveToFile(const Student* students, size_t count,
                                    ofstream& file)
 {
    file.write((const char*) students, count*sizeof(Student));
 }
size_t getFileSize(ifstream& f)
 { size_t currPos = f.tellg();
     f.seekg(0, ios::end);
    size_t size = f.tellg();
    f.seekg(0, currPos);
     return size;
 }
```

[illegible]

```
void readFromFile(Student*& ptr, size_t& studentsCou... #
                  ifstream& f)
{ size_t sizeOfFile = getFileSize(f);
  studentsCount = sizeOfFile/sizeof(Student);

  ptr= new Student[studentsCount];
  f.read((char*)ptr, sizeOfFile);
}

int main()
{ {
  Student* arr = new Student[3];
  initStudent(arr[0], "ivan, 44, 1234);
  initStudent(arr[1], "petur", 12, 765);
  initStudent(arr[2], "alex", 15, 44);

  ostream file("students.dat", ios::binary);
  if(!file.is_open())
  {
      ~~std~~ cout << "Eror while opening the fil!" <<endl;
      delete[] arr;
      return -1;
  }
  saveToFile(arr, ~~size~~ 3, file);
  delete[] arr;
  return 0;
} }
  { Student* arr;
  size_t count;
  ifstream file("students.dat", ios::binary);
  readFromFile(arr, count, file); delete[] arr; }}
```

ArrayDynamicNameStudent
.cpp

```
include <iostream>
# include <fstream>
#pragma warning (disable:4996)
~~#~~using namespace std;
struct DynamicName Student
{
    char* name;
    int age;
    int fn;
};

DynamicName Student init(const char* name, int age, int fn)
 {
    DynamicName Student st;
    st.name = new char[strlen(name)+1];
    strcpy(st.name, name);
    st.age = age;
    st.fn = fn;
    return st;
}
void freeStudent(DynamicNameStudent& st)
 { delete[] st.name;
   st.age = st.fn = 0;
}
void saveToBinaryFile(ofstream& ofs, const DynamicNameSt& st)
{ int nameLen = strlen(st.name);
  ofs.write((const char*) &nameLen, sizeof(int));
  Ofs.write(st.name, nameLen+1);
  ofs.write((const char*)& st.age, sizeof(st.age));
} ofs.write((const char*)&st.fn, sizeof(st.fn));
```

```
DynamicNameStudent readFromBinFile(ifstream& ifs)
{ DynamicNameStudent st2;
  int nameLen;
  ifs.read((char*)& nameLen, sizeof(nameLen));
  st2.name=new char[nameLen+1];
  ifs.read(st2.name, nameLen+1);
  ifs.read((char*)&st2.age, sizeof(st2.age));
  ifs.read((char*)& st2.fn, sizeof(st2.fn));
  return st2;
}
void saveStudentsArrayToFile(ofstream& ofs, const
                              DynamicNameSt* arr,
                              size_t arraySize)
{ ofs.write((const char*)& arraySize, sizeof(arraySize));
  for(size_t i=0;i<arraySize;i++)
      saveToBinaryFile(ofs, arr[i]);
}
void readStudents(ifstream& ifs, DynamicNameStudent*& ptr,
                  size_t& size)
{ ifs.read((char*)& size, sizeof(size));
   ptr=new DynamicNameSt[size];
   for(size_t i=0; i<size;i++)
       ptr[i]=readFromBinaryFile(ifs);
}
```

ArrayDynamic
N 02 NameStud.cpp

```
int main()
{
  {
    ofstream ofs("arrStudents.dat",
                               std::ios::binary);

    if(!ofs.is_open())
          return -1;
    DynamicNameStudent arr[3];
    arr[0] = init("Ivan", 10, 20);
    arr[1] = init("Peter", 41, 44);
    arr[2] = init("OK", 99, 0);

    saveStudentsArrayToFile(ofs, arr, 3);
    for(size_t i=0; i<3; i++)
          freeStud(arr[i]);

  }

  {
    ifstream ifs("arrStudents.dat", std::ios::binary);
    if(!ifs.is_open())
          return -1;
    DynamicNameStudent* arr;
    size_t size;
    readStuds(ifs, arr, size);

    for(size_t i=0; i<size; i++)
          freeStuds(arr[i]);
  }
}
```