Тема 03. Двоични файлове и fstream

01. Двоичен файл

def| готов за зареждане в паметта **(файлове с пряк достъп)**

Двоичният файл записва информацията точно като е представена в паметта.

За да направим разликата, нека имаме числото 5.
При **стандартния файл**, той ще запише **"5"**. Докато при **двоичния** ще се запазят байтовете на числото 5, в случая **[0] [0] [0] [5]**, ако числото е от тип **int**.

Тестов файл - лесен за нас, труден за програмата
Двоичен файл - труден за нас, лесен за програмата

```
//познатата ни операция за изход при стандартните файлове <<,
//тук е .write(), а операцията за вход >>, тук е .read()

std::fstream inFile("text.txt", std::ios::binary);

int a = 5;
inFile.write((const char*)&a, sizeof(a));

//              ^                ^
//              |                |
//превръщаме обекта,           броят байтове, които
//който искаме да запишем      искаме да запишем
//CONST char масив, тъй като
//char е 1 байт

char buff[10];
inFile.read(buff, 10); // <- броят байтове, които искаме да запишем
//           ^
//           |
//превръщаме обекта,
//в който искаме да запишем
//в char масив, тъй като
//char е 1 байт
```

```
//-> структури с "външен ресурс"
struct Person
{
    char* name = nullptr;
    int age;
}

//тук, name е пойнтър към char, което означава, че може да сочи към
//низ от символи (string) съхраняван някъде в паметта.
//Това е типичен пример за "външен ресурс" - памет,
//която не е част от самата структура Person,
//но е от съществено значение за нейната коректна функционалност.

std::cout << sizeof(Person); //ще включи паметта отделена за ПОЙНТЪРА,
                             //а не самия масив

write((...)&p, sizeof(P)); //така ще запишем указателя (адрес)
                           //и годините, което губи ползите от двоичния файл

//=> [nameSize] [name] [age], ще е нашето съдържание на файла,
      //т.е. запазваме дължината на името, самото име и годините
```

!!! Указателят е един **put** и **get** са един и същи указатели **(не винаги е така)**

```
std::fstream file(fileName);

std::cout<< tellp() == tellg() <<std::endl; // 1 (true)

//важно е да кажем, че ако преместим единия, то се мести и другия, т.е
file.seekg(1, std::ios::cur);

std::cout<< tellp() == tellg() <<std::endl; // 1 (true)

//При входна операция след изходна трябва да синхронизираме буфера

file << "a";
file.flush(); //синхронизация

//или seekg(tellg());

int num = 0;
file >> num;
```