Тема 3 * Да премести fstream

Tuesday, July 2, 2024 4:22 PM

Тема 3

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. Пример за запазване/четене на масив от обекти(от един тип) във файл. Пример за заместване на символ с друг символ във файл (четене и писане с fstream)

Двоични файлове - файл, готов за зареждане в паметта

- разликата с текстовите файлове е в интерпретацията (нема как да разберем дали един файл е текстов или двоичен)
- предимството им е, че работата с тях е по-лесна за програмата, защото нямаме преобразуването от стринг в число, които имаме, когато се зареждаме текстов файл в паметта
- режим на работа с двоични файлове
  std::ios::binary

1. Запазване в двоичен файл

ofs.write(const char*, size(колко елемента байтове да запише))

когато искаме да запазим променливи, които не е char, трябва експлицитно с кастване към const char*

а) Запазване на примитивни типове

```
int x = 5;
ofs.write((const char*)&x , sizeof(int));
```

б) Запазване на структури

```
struct Test1{          без динамична памет
    bool b=0;
    int x = 0;
};

int main{
  Test1 t = {true, 45};
    ofs.write((const char*)&t, sizeof(t));
}

struct student{        с динамична памет
    char* name;
    int fn;
};
```

- изнасяме запазването във

```
void saveStudentToFile (std::ofstream& ofs, const Student& t)
```

- изнасяме запазването във ф-я, чрез която запазваме поле на масива и всички член-данни поотделно

```
void saveStudentToFile(std::ofstream& ofs, const Student& t)
    size_t namelen = strlen(st.name);
    ofs.write((const char*)&namelen, sizeof(namelen)); //първо запазване големината на масива
    ofs.write(st.name, namelen);
    ofs.write((const char*)& fn, sizeof(fn));
}
```

в) Запазване на масив от обекти

без динамична памет

```
struct Student{
    char name[30];
    int fn;
};
```

- правим функция, в която големината на масива се подава като пар-р

```
void saveToFile(const Student* st, std::ofstream& ofs, size_t count) {
    ofs.write((const char*) st, count * sizeof(student));
}
```

2. Четене от двоичен файл

`ifs.rea(char* buffer, size_t)`

char* buffer – ук-л към място, на което да се поставят данните

size_t – колко байта да се прочетат

а)
```
int a;
ifs.read((char*)& a, sizeof(int));
```

б) без дин. памет
```
Test1 t;
ifs.read((char*) t, sizeof(t));
```

с дин. памет
```
struct Student {
    char* name;
    int fn;
};

Student readStudentFromFile(ifstream& ifs) {
    Student res;
```

```
    size_t nameLen=0;
    ifs.read((char*)nameLen, sizeof(nameLen)); - първо прочитаме
                                               дължината и масива !!!
    res.name= new char[nameLen+1];
    ifs.read(res.name, nameLen+1);
    ifs.read((char*)&res.fn, sizeof(res.fn));
    return res;
}
```

б)

```
struct Student{
    char name[30];
    int fn;
}

size_t getFileSize(){...};

void readStudentFromFile(Student*&ptr, size_t& studentsCount, ifstream ifs)
{
    size_t fileSize=getFileSize(ifs);
    studentsCount = fileSize /sizeof(Student);
    ptr=new Student[studentsCount];
    ifs.read((char*)ptr, fileSize);
}

int main(){
    Student* arr;
    size_t studentsCount;
    ⋮
    readStudentFromFile(arr, studentsCount, ifs);
    delete[] arr;
```

3. Съдържание на файла - шестнадесетичното представяне на данните

- байтовете се записват отзад напред (най-старшият байт е последен)

```
struct Test{
    char ch;
```

```
struct Test {
    char ch;
    int a;
};

int main() {
    Test t[3] = {{'a', 400}, {'b', 500}, {'c', 600}};
    f.write((const char*)&t, sizeof(arr));
}
```