2023 12 15

Dosya İşlemleri

```
std::ifstream open_binary_file(const std::string& filename)
{
    std::ifstream ifs {filename, std::ios::binary};
    if (!ifs)
    {
        throw std::runtime_error{ filename + " dosyayisi acilamiyor\n"};
    }
    return ifs;
}

int main()
{
    using namespace std;
    constexpr size_t n = 10'000u;

    int x{};
    int prime_count{};

    auto ofs = create_binary_file("primes10000.dat");

    while (prime_count < n)
    {
        if (isprime(x))
        {
            ofs.write(reinterpret_cast<char*>(&x), sizeof(int));
            ++prime_count;
        }
        ++x;
    }

    ofs.close();
}
```

```
int main()
{
    auto ifs = open_binary_file("primes10000.dat");

    int x{};

    while (ifs.read(reinterpret_cast<char*>(&x), sizeof(x)))
    {
        cout << x << "\n";
    }

    vector<int> ivec(10'000);

    ifs.read(reinterpret_cast<char*>(ivec.data(), 10'000 * sizeof(int)));
}
```

```cpp
// bir dosyayı parçalara bölme
int main(int argc, char* argv[])
{
    using namespace std;

    if (argc != 3)
    {
        cerr << "kullanim: <dbol> <dosya ismi> <byte sayisi>\n";
        return 1;
    }

    char c{};
    int file_count{};
    int byte_count{};

    auto ifs = open_binary_file(argv[1]);

    int chunk = atoi(argv[2]);
    ofstream ofs;
    ostringstream ostr;
    ostr.fill('0');

    while (ifs.get(c))
    {
        if (!ofs.is_open())
        {
            ostr << "parca" << setw(3) << file_count + 1 << ".par";
            ofs.open(ostr.str(), ios::binary);

            if (!ofs)
            {
                cerr << ostr.str() << "dosya olusturulamadi\n";
                return 1;
            }
            ++file_count;
        }
        ofs.put(c);
        ++byte_count;
        if (byte_count % chunk == 0)
        {
            ofs.close();
        }
    }
}
```

```cpp
int main(int argc, char **argv)
{
    using namespace std;

    if (argc != 2)
    {
        cerr << "kullanim: <dbir> <yeni dosya ismi>\n";
        return 1;
    }

    auto ofs = create_binary_file(argv[1]);

    int file_count{};

    for(;;)
    {
        ostringstream ostr;
        ostr.fill('0');
        ostr << "parca" << setw(3) << file_count + 1 << ".par";

        ifstream ifs{ ostr.str(), ios::binary };

        if (!ifs)
                break;

        char c;
        while (ifs.get(c))
        {
            ofs.put(c);
            ++byte_count;
        }
        ++file_count;
        ifs.close();

        if (remove(ostr.str().c_str())
        {
            cerr << "dosya silinemedi\n";
            return 2;
        }
    }
}
```

```cpp
// dosya konum göstericisi
/*
    seekg okuma amaçlı konumlandırma     --ifstream
    seekp yazma amaçlı konumlandırma     --ofstream

    tellg   okuma konunumu döndürür      --ifstream
    tellp   yazma konumunu döndürür      --ofstream

    ios::beg konumlandırma dosyanın başından yapılmak için
    ios::end konumlandırmanın dosyanın sonunda yapılması için
    ios::cur dosya konum gösterisinin son konumdan gösterim yapmak için
    (mevcut konumu döndürür)
*/
```

```cpp
#include <sstream>
#include <iostream>
#include <iomanip>
int main()
{
    using namespace std;

    istringstream iss{"necati ergin"};

    string str;
    iss >> str;

    cout << quoted(str) << "\n"; // "necati"
    iss.seekg(0); // iss.seekg(0, ios::beg) farkı yok
    iss >> str;
    cout << quoted(str) << "\n"; // "necati"


    iss.seekg(1, ios::cur);
    iss >> str;
    cout << quoted(str) << "\n"; // "ergin"

    iss.seekg(-6, ios::cur);
    iss >> str;
    cout << quoted(str) << "\n";  // "ergin"

    iss.seekg(-6, ios::end);
    iss >> str;
    cout << quoted(str) << "\n";  // "ergin"
}
```

```cpp
int main()
{
    using namespace std;

    ostringstream oss{"furkan mert"};

    cout << "[" << oss.str() << "]\n"; // [furkan mert]

    oss.seekp(3);
    oss.put(!);
    cout << "[" << oss.str() << "]\n"; // [fur!an mert]

    oss.seekp(0);
    oss.put('*');
    cout << "[" << oss.str() << "]\n"; // [*ur!an mert]
}
```

```cpp
int main()
{
    using namespace std;

    auto ifs = open_binary_file("primesmillion.dat");
    int n{};

    std::cout << "kacinci asal sayi : ";
    cin >> n;

    ifs.seekg((n -1) * sizeof(int), ios::beg);

    int x;
    ifs.read(reinterpret_cast<char*>(&x), sizeof(int));

    cout << n << ".asal sayi " << x << "\n";
}
```

```cpp
void print_file(const std::string& filename, int ntimes)
{
    auto ifs = open_text_file(filename);

    while (ntimes--)
    {
        std::cout << ifs.rdbuf();
        (void)getchar();
        ifs.seekg(0);
    }
}

int main()
{
    print_file("main.cpp", 5); // 5 kez dosyayı yazacak
}
```

```cpp
// gcount()
int main()
{
    int *p = new int[20000];
    auto ifs = open_binary_file("primes10000.dat");

    ifs.read(reinterpret_cast<char*>(p), sizeof(int) * 20000);
    // 20klık int değeri dolduramadı o yüzden stream hata durumunda oldu
    if (ifs)
    {
        std::cout << "stream hata durumunda degil\n";
    }
    else
    {
        std::cout << "stream hata durumunda\n";
    }

    // okunan byte sayisi 40'000
    std::cout << "okunan byte sayisi : " << ifs.gcount() << "\n";

    delete p;
}
```