Практика

1 курс

а) Формирование исходных бинарных файлов на базе исходных текстовых

```
74  void createBinaryStudents(const std::string& inputFile, const std::string& outputFile) {
75      std::ifstream in(inputFile);
76      std::ofstream out(outputFile, std::ios::binary);
77
78      std::string line;
79      while (std::getline(in, line)) {
80          std::stringstream ss(line);
81          std::string inlexStr, lastNameStr, firstNameStr, patronymicStr;
82
83          if (std::getline(ss, inlexStr, ';') &&
84              std::getline(ss, lastNameStr, ';') &&
85              std::getline(ss, firstNameStr, ';') &&
86              std::getline(ss, patronymicStr, ';')) {
87
88              StudentFull s{};
89
90
91              s.id = stoi(inlexStr);
92
93              std::strncpy(s.lastName, lastNameStr.c_str(), sizeof(s.lastName));
94
95              std::strncpy(s.firstName, firstNameStr.c_str(), sizeof(s.firstName));
96
97              std::strncpy(s.patronymic, patronymicStr.c_str(), sizeof(s.patronymic));
98
99              s.group = 0;
100             s.math = s.geo = s.prog = 0;
101             s.average = 0.0;
102
103             out.write(reinterpret_cast<char*>(&s), sizeof(StudentFull));
104         }
105     }
106
107     in.close();
108     out.close();
109 }
```

Задача функции createBinaryStudents создать бинарный файл с данными о студентах из текстового файла

Описание работы функции

1) Открытие файлов.

   Открывает входной файл для чтения.

   Открывает выходной файл для записи в бинарном формате.

```
std::ifstream in(inputFile);
std::ofstream out(outputFile, std::ios::binary);
```

2) Чтение данных.

   Считывает строки из входного файла до конца файла.

```
std::string line;
while (std::getline(in, line)) {
```

3) Парсинг данных

Для каждой строки:

Создает строковый поток (std::stringstream) для разбора.

Извлекает поля: индекс, фамилию, имя и отчество.

```
std::stringstream ss(line);
std::string inlexStr, lastNameStr, firstNameStr, patronymicStr;
```

4) Проверка и создание структуры

Проверяет, успешно ли извлечены все поля.

Создает объект структуры StudentFull для хранения информации о студенте

```
if (std::getline(ss, inlexStr, ';') &&
    std::getline(ss, lastNameStr, ';') &&
    std::getline(ss, firstNameStr, ';') &&
    std::getline(ss, patronymicStr, ';')) {

    StudentFull s{};

    s.id = stoi(inlexStr);
```

5) Копирование данных

Копирует фамилию, имя и отчество в соответствующие поля структуры.

```
std::strncpy(s.lastName, lastNameStr.c_str(), sizeof(s.lastName));
std::strncpy(s.firstName, firstNameStr.c_str(), sizeof(s.firstName));
std::strncpy(s.patronymic, patronymicStr.c_str(), sizeof(s.patronymic));
```

6) Инициализация остальных полей

Устанавливает значения для группы и оценок в 0, а среднее значение в 0.0.

```
s.group = 0;
s.math = s.geo = s.prog = 0;
s.average = 0.0;
```

7) Запись в бинарный файл. Закрытие файлов

Записывает объект StudentFull в выходной бинарный файл.
Закрывает входной и выходной файлы после завершения работы

```
out.write(reinterpret_cast<char*>(&s), sizeof(StudentFull));
in.close();
out.close();
```

b) Подсоединения фамилии студентов к ведомостям оценок с формированием нового бинарного файла

```cpp
void createBinaryGrades(const std::string& inputFile, const std::string& outputFile) {
    std::ifstream in(inputFile);
    std::ofstream out(outputFile, std::ios::binary);
    std::string line;
    while (std::getline(in, line)) {
        std::stringstream ss(line);
        std::string group, index;
        std::string subj1, grade1, subj2, grade2, subj3, grade3;

        if (std::getline(ss, group, ';') &&
            std::getline(ss, index, ';') &&
            std::getline(ss, subj1, ';') &&
            std::getline(ss, grade1, ';') &&
            std::getline(ss, subj2, ';') &&
            std::getline(ss, grade2, ';') &&
            std::getline(ss, subj3, ';') &&
            std::getline(ss, grade3, ';')) {

            StudentFull s{};
            s.group = stoi(group);
            s.id = stoi(index);
            s.math = stoi(grade1);
            s.geo = stoi(grade2);
            s.prog = stoi(grade3);

            out.write(reinterpret_cast<char*>(&s), sizeof(StudentFull));
        }
    }

    in.close();
    out.close();
}
```

Задача данной функции создать бинарный файл, в котором будут данные студентов и их оценки

Описание работы функции

1) Открытие файлов
2) Чтение данных
3) Парсинг данных

Для каждой строки: Создает строковый поток (std::stringstream) для разбора.

Извлекает поля: группу, индекс студента, предметы и оценки.

```
std::stringstream ss(line);
std::string group, index;
std::string subj1, grade1, subj2, grade2, subj3, grade3;
```

4) Проверка и создание структуры

Проверяет, успешно ли извлечены все поля.

Создает объект структуры StudentFull для хранения информации о оценках студента.

```
if (std::getline(ss, group, ';') &&
    std::getline(ss, index, ';') &&
    std::getline(ss, subj1, ';') &&
    std::getline(ss, grade1, ';') &&
    std::getline(ss, subj2, ';') &&
    std::getline(ss, grade2, ';') &&
    std::getline(ss, subj3, ';') &&
    std::getline(ss, grade3, ';')) {

    StudentFull s{};
    s.group = stoi(group);
```

5) Заполнение оценок

Преобразует оценки из строк в целые числа и заполняет соответствующие поля структуры.

```
s.id = stoi(index);
s.math = stoi(grade1);
s.geo = stoi(grade2);
s.prog = stoi(grade3);
```

6) Запись в бинарный файл. Закрытие файлов

c) Вычисление среднего балла каждого студента с формированием нового бинарного файла;

```cpp
void calculateAverage(const std::string& inputBin, const std::string& outputBin) {
    std::ifstream in(inputBin, std::ios::binary);
    std::ofstream out(outputBin, std::ios::binary);

    StudentFull s{};
    while (in.read(reinterpret_cast<char*>(&s), sizeof(StudentFull))) {

        StudentFull avg{};

        avg.id = s.id;
        avg.group = s.group;
        strncpy(avg.lastName, s.lastName, sizeof(avg.lastName));

        avg.math = s.math;
        avg.geo = s.geo;
        avg.prog = s.prog;
        avg.average = (s.math + s.geo + s.prog) / 3.0;
        out.write(reinterpret_cast<char*>(&avg), sizeof(StudentFull));
    }
}
```

Функция производит вычисление средних оценок студентов и запись результатов в новый бинарный файл.

Описание работы функции

1) Открытие файлов
2) Объявление структуры

```
StudentFull s{};
```

3) Чтение данных
4) Создание объекта для хранения данных

Создает новый объект avg для записи информации о студ

```
StudentFull avg{};
```

5) Копирование данных

```
avg.id = s.id;
avg.group = s.group;
strncpy(avg.lastName, s.lastName, sizeof(avg.lastName));

avg.math = s.math;
avg.geo = s.geo;
avg.prog = s.prog;
avg.average = (s.math + s.geo + s.prog) / 3.0;
```

6) Вычисление среднего

Вычисляет среднюю оценку на основе трех предметов и записывает её в поле average.

```
avg.average = (s.math + s.geo + s.prog) / 3.0;
```

7) Запись в выходной файл

d) Формирование списка неуспевающих, состоящего из фамилии, номера группы, номера зачетки

```
void createFailingList(const std::string& averageBin, const std::string& failingBin) {
    std::ifstream in(averageBin, std::ios::binary);
    std::ofstream out(failingBin, std::ios::binary);

    StudentFull s{};

    while (in.read(reinterpret_cast<char*>(&s), sizeof(StudentFull))) {
        if (s.average < 4.0) {

            StudentFull fs{};

            fs.id = s.id;
            fs.group = s.group;
            strncpy(fs.lastName, s.lastName, sizeof(fs.lastName));
            fs.average = s.average;
            fs.math = s.math;
            fs.geo = s.geo;
            fs.prog = s.prog;

            out.write(reinterpret_cast<char*>(&fs), sizeof(StudentFull));
        }
    }
}
```

Функция создает список неуспевающих студентов и записывает данные в бинарный файл

Описание работы функции

1) Открытие файлов
2) Объявление структуры

```
StudentFull s{};
```

3) Чтение данных

```
while (in.read(reinterpret_cast<char*>(&s), sizeof(StudentFull))) {
```

4) Проверка на неуспех

Проверяет, ниже ли средняя оценка 4.0.

```
if (s.average < 4.0) {
```

5) Создание объекта для неуспевающего студента

Создает новый объект fs для записи информации о неуспевающем студенте.

```
StudentFull fs{};
```

6) Заполнение полей структуры

Заполняет поля нового объекта данными о студе

```
fs.id = s.id;
fs.group = s.group;
strncpy(fs.lastName, s.lastName, sizeof(fs.lastName));
fs.average = s.average;
fs.math = s.math;
fs.geo = s.geo;
fs.prog = s.prog;
```

7) Запись в выходной файл

```
out.write(reinterpret_cast<char*>(&fs), sizeof(StudentFull));
```

е) Сортировка списка неуспевающих по группам, в группе - по фамилиям в алфавитном порядке

```
void sortFailingList(const std::string& failingBin) {
    std::ifstream in(failingBin, std::ios::binary);
    StudentFull s{};
    int size{};

    StudentFull* arr = nullptr;

    while(in.read(reinterpret_cast<char*>(&s), sizeof(StudentFull))) {
        addElementToArray(arr, size, s);
    }

    std::sort(arr, arr + size, compareGroupAndSurnames);

    std::ofstream out(failingBin, std::ios::binary | std::ios::trunc);


    for (int i = 0; i < size; ++i) {
        out.write(reinterpret_cast<char*>(&arr[i]), sizeof(StudentFull));
    }

    delete[] arr;
}
```

Функция производит сортировку списка неуспевающих студентов по группам и фамилиям, а затем запись отсортированных данных обратно в бинарный файл.

Описание работы функции

1) Открытие файла
2) Объявление переменных

```
StudentFull s{};
int size{};
StudentFull* arr = nullptr;
```

3) Чтение данных о неуспевающих студентах и запись их в массив

```
while(in.read(reinterpret_cast<char*>(&s), sizeof(StudentFull))) {
    addElementToArray(arr, size, s);
}
```

4) Сортировка данных

Использует стандартный алгоритм сортировки для упорядочивания студентов по группам и фамилиям с помощью функции compareGroupAndSurnames.

```
std::sort(arr, arr + size, compareGroupAndSurnames);
```

```
bool compareGroupAndSurnames(const StudentFull& a, const StudentFull& b) {

    if (a.group != b.group)
        return a.group < b.group;

    return std::strcmp(a.lastName, b.lastName) < 0;
}
```

5) Запись в выходной файл. Очистка памяти

```
std::ofstream out(failingBin, std::ios::binary | std::ios::trunc);
for (int i = 0; i < size; ++i) {
    out.write(reinterpret_cast<char*>(&arr[i]), sizeof(StudentFull));
}
delete[] arr;
```

f) Распечатка бинарного файла до сортировки и после

```
void printFailingList(const std::string& failingBin) {
    std::ifstream in(failingBin, std::ios::binary);
    StudentFull fs{};

    while (in.read(reinterpret_cast<char*>(&fs), sizeof(StudentFull))) {
        std::cout << "Last Name: " << fs.lastName
            << ", Group: " << fs.group
            << ", ID: " << fs.id
            << ", Average Grade: " << std::fixed << std::setprecision(4) << fs.average
            << std::endl;
    }
}
```

Функция производит вывод на консоль данных из бинарного файл

Описание работы функции

1) Открытие файла
2) Использование объекта

Использование объекта с данными о неуспевающем студенте

```
StudentFull fs{};
```

3) Чтение данных

В цикле while происходит считывание записей студентов из файла. Чтение продолжается до тех пор, пока не достигнут конец файла.

```
while (in.read(reinterpret_cast<char*>(&fs), sizeof(StudentFull))) {
```

4) Вывод на консоль

Для каждого считанного студента выводится следующая

информация: Фамилия, Группа, ID, Средняя оцен

```
std::cout << "Last Name: " << fs.lastName
    << ", Group: " << fs.group
    << ", ID: " << fs.id
    << ", Average Grade: " << std::fixed << std::setprecision(4) << fs.average
    << std::endl;
```

g) Формирование ведомости оценок для заданной группы, упорядоченной по алфавиту

```
void printGroupSortedBySurnames(const std::string& mergedBin, int targetGroup) {
    std::ifstream in(mergedBin, std::ios::binary);

    StudentFull* arr = nullptr;
    int size{};
    StudentFull s{};


    while (in.read(reinterpret_cast<char*>(&s), sizeof(StudentFull))) {
        if (s.group == targetGroup) {
            addElementToArray(arr, size, s);
        }
    }

    in.close();

    std::sort(arr, arr + size, compareSurnames);

    std::ofstream out(mergedBin, std::ios::binary);

    for (int i = 0; i < size; ++i) {
        out.write(reinterpret_cast<char*>(&arr[i]), sizeof(StudentFull));
    }

    out.close();

    delete[] arr;
}
```

Функция предназначена для сортировки студентов по фамилиям в указанной группе и записи отсортированных данных обратно в бинарный файл.

Описание работы функции

1) Открытие файла
2) Использование динамического массива

Создаётся указатель на массив StudentFull для хранения студентов целевой группы, а также переменная size для отслеживания количества студентов.

```
StudentFull* arr = nullptr;
int size{};
StudentFull s{};
```

3) Чтение данных и запись в массив. Закрытие файла

В цикле while происходит считывание записей студентов из файла. Если группа студента совпадает с целевой, запись добавляется в массив.

```
while (in.read(reinterpret_cast<char*>(&s), sizeof(StudentFull))) {
    if (s.group == targetGroup) {
        addElementToArray(arr, size, s);
    }
}
in.close();
```

4) Сортировка массива

После завершения чтения данные студентов целевой группы сортируются по фамилиям с помощью функции std::sort и функции сравнения compareSurnames.

```
std::sort(arr, arr + size, compareSurnames);

bool compareSurnames(const StudentFull& a, const StudentFull& b) {

    return std::strcmp(a.lastName, b.lastName) < 0;
}
```

5) Запись данных в файл

Открывается тот же бинарный файл для записи. Отсортированные данные записываются обратно в фа

```
std::ofstream out(mergedBin, std::ios::binary);

for (int i = 0; i < size; ++i) {
    out.write(reinterpret_cast<char*>(&arr[i]), sizeof(StudentFull));
}

out.close();
```

6) Очистка памяти

h) Формирование ведомости оценок для заданной группы, упорядоченной по убыванию среднего балла

```
void printGroupSortedByAverage(const std::string& mergedBin, int targetGroup) {
    std::ifstream in(mergedBin, std::ios::binary);

    StudentFull* arr = nullptr;
    int size{};
    StudentFull s{};


    while (in.read(reinterpret_cast<char*>(&s), sizeof(StudentFull))) {
        if (s.group == targetGroup) {
            addElementToArray(arr, size, s);
        }
    }

    in.close();

    std::sort(arr, arr + size, compareAverage);

    std::ofstream out(mergedBin, std::ios::binary);

    for (int i = 0; i < size; ++i) {
        out.write(reinterpret_cast<char*>(&arr[i]), sizeof(StudentFull));
    }

    out.close();

    delete[] arr;
}
```

Функция предназначена для сортировки студентов по убыванию среднего балла в указанной группе и записи отсортированных данных обратно в бинарный файл.

Описание работы функции

1) Открытие файла
2) Использование динамического массива

Создаётся указатель на массив StudentFull для хранения студентов целевой группы, а также переменная size для отслеживания количества студентов.

```
StudentFull* arr = nullptr;
int size{};
StudentFull s{};
```

3) Чтение данных и запись в массив. Закрытие файла

В цикле while происходит считывание записей студентов из файла. Если группа студента совпадает с целевой, запись добавляется в массив.

```
while (in.read(reinterpret_cast<char*>(&s), sizeof(StudentFull))) {
    if (s.group == targetGroup) {
        addElementToArray(arr, size, s);
    }
}
in.close();
```

4) Сортировка массива

После завершения чтения данные студентов целевой группы сортируются по фамилиям с помощью функции std::sort и функции сравнения compareAverage.

```
std::sort(arr, arr + size, compareAverage);
```

```
bool compareAverage(const StudentFull& a, const StudentFull& b) {

    return a.average > b.average;
}
```

i) Формирования списка отличников, состоящего из фамилии, номера группы, номера зачетки

```
void createGoodList(const std::string& inBin, const std::string& outBin) {
    std::ifstream in(inBin, std::ios::binary);
    std::ofstream out(outBin, std::ios::binary);

    StudentFull s{};
    while (in.read(reinterpret_cast<char*>(&s), sizeof(StudentFull))) {
        if (s.average >= 8.0) {

            StudentFull gs{};

            strncpy(gs.lastName, s.lastName, sizeof(gs.lastName));
            gs.id = s.id;
            gs.group = s.group;
            gs.average = 0.0;
            gs.math = 0;
            gs.geo = 0;
            gs.prog = 0;


            out.write(reinterpret_cast<char*>(&gs), sizeof(StudentFull));
        }
    }
}
```

Функция предназначена для создания списка студентов с хорошими оценками (средняя оценка 8.0 и выше) и записи этих данных в новый бинарный файл.

Описание работы функции

1) Открытие файлов
2) Чтение данных

В цикле while происходит считывание записей студентов из входного файла.

```
while (in.read(reinterpret_cast<char*>(&s), sizeof(StudentFull))) {
```

3) Отбор студентов. Запись данных

Если средняя оценка студента равна или превышает 8.0, создаётся новый объект StudentFull, в который копируются данные о фамилии, ID и группе. Остальные поля инициализируются нулями.

```
if (s.average >= 8.0) {

    StudentFull gs{};

    strncpy(gs.lastName, s.lastName, sizeof(gs.lastName));
    gs.id = s.id;
    gs.group = s.group;
    gs.average = 0.0;
    gs.math = 0;
    gs.geo = 0;
    gs.prog = 0;


    out.write(reinterpret_cast<char*>(&gs), sizeof(StudentFull));
}
```