Тест[1] для соискателей вакансии
Программист C++

Уважаемый соискатель, перед Вами тестовое задание на вакансию «Программист C++» в компании Saber Interactive.

Индустрия видеоигр, в которой работает компания Saber, предъявляет высокие требования к качеству, эффективности и удобству восприятия программного кода. Поэтому мы предлагаем выполнить Вам три задания, чтобы Вы могли продемонстрировать Вашу способность писать такой код.

Просьба не использовать в первых двух задачах библиотечные функции и классы, являющиеся решением этих задач (например, класс std::bitset, являющийся решением для первой задачи). Для третьей задачи можно использовать любые функции и контейнеры из STL.

После выполнения задания в комментарии к коду укажите, пожалуйста:

Фамилию, имя, отчество
Дату выполнения и примерное количество времени, затраченного на выполнение теста.
Готовое решение пришлите, пожалуйста, в cpp.

Техническое задание:

1. Напишите функцию, которая принимает на вход знаковое целое число и печатает его двоичное представление.

2. Напишите функцию, удаляющую последовательно дублирующиеся символы в строке:

```
void RemoveDups(char* str);

// пример использования
char data[] = "AAA BBB   AAA";
RemoveDups(data);
printf("%s\n", data); // "A B A"
```

1

[?]Данное задание разработано специально для тестирования кандидатов и не имеет отношения к какому-либо конкретному проекту, поэтому мы гарантируем, что ваш концепт арт, рисунок, код, модель и т.д., выполненный в качестве теста не будет использоваться в коммерческой разработке.

3. Реализуйте функции сериализации и десериализации двусвязного списка. Данные должны быть записаны в бинарном формате. Ожидаемая алгоритмическая сложность – меньше квадратичной.

```
// структуру ListNode модифицировать нельзя
struct ListNode {
    ListNode* prev = nullptr; // указатель на предыдущий элемент
списка, либо `nullptr` в случае начала списка
    ListNode* next = nullptr;
    ListNode* rand = nullptr; // указатель на произвольный элемент
данного списка, либо `nullptr`
    std::string data; // произвольные пользовательские данные
};

class List {
public:
    void Serialize(FILE* file); // сохранение списка в файл, файл
открыт с помощью `fopen(path, "wb")`
    void Deserialize(FILE* file); // восстановление списка из файла,
файл открыт с помощью `fopen(path, "rb")`

    // ... ваши методы для заполнения списка

private:
    ListNode* head = nullptr;
    ListNode* tail = nullptr;
    int count = 0;
};
```

Примечание: сериализация подразумевает сохранение и восстановление полной структуры списка, включая взаимное соотношение его элементов между собой.

Спасибо за уделенное время и выполнение нашего теста!