Тест[1] для соискателей вакансии
Программист графики

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

Просьба не использовать в первой задаче библиотечные функции и классы, являющиеся решением этих задач (например stl::bitset). Для второй и третьей задачи можно использовать стандартные функции и контейнеры из STL.

После выполнения задания в комментарии к коду укажите, пожалуйста:

Дату выполнения и примерное количество времени, затраченного на выполнение теста.
Готовое решение, пожалуйста, прикрепите по изначальной ссылке zip-архивом.

Техническое задание:

1. Напишите функцию, которая принимает на вход беззнаковое целое число и производит перестановку бит в обратном порядке в принимаемом значении, то есть производит перестановки по битам пар: 0-31, 1-30, 2-29 и т. д, а затем возвращает результат этой перестановки. Считается, что длина битовой последовательности всегда равна 32.

---

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. Конечную работу вы можете использовать в своем портфолио и размещать на любых ресурсах.

2. Реализуйте функции сериализации и десериализации двусвязного списка в бинарном формате в файл. Алгоритмическая сложность решения должна быть меньше квадратичной.

```
// структуру ListNode модифицировать нельзя
struct ListNode {
    ListNode * prev;
    ListNode * next;
    ListNode * rand; // указатель на произвольный элемент данного списка,
либо NULL
    std::string data;
};
class List {
public:
    void Serialize (FILE * file); // сохранение в файл (файл открыт с
помощью fopen(path, "wb"))
    void Deserialize (FILE * file); // загрузка из файла (файл открыт с
помощью fopen(path, "rb"))

private:
    List Node * head;
    List Node * tail;
    int count;
};
```

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

3. Реализуйте функцию расчета усредненных нормалей для вершин меша, заданного индексированными треугольниками.

```
//
// Calculate smooth (average) per-vertex normal
//
// [out] normals - output per-vertex normal
// [in] verts - input per-vertex positions
// [in] faces - triangles (triplets of vertex indices)
// [in] nverts - total number of vertices
// [in] nfaces - total number of triangles
//
void calc_mesh_normals(vec3* normals, const vec3* verts, const int* faces,
int nverts, int nfaces)
{
}
```

Спасибо за уделенное время и выполнение нашего теста!