```cpp
#include <vector>
#include <string>
#include "QPanda.h"
using namespace std;
USING_QPANDA
QCircuit amplitude_encode(qvec q, vector<double> data);
QCircuit init_superposition_state(qvec q, size_t d);

class QSolver
{
public:
    size_t m_grid_number;
    //j_0=sqrt(blog(4b/epsilon))
    size_t Cheby_times;
    //b=(kappa*d)^2*log(kappa*d/epsilon)
    size_t m_b;
    size_t m_sparse_coef;
    vector<double> m_alpha;
    QCircuit T_circuit(qvec qi, qvec qi_anc, qvec qj, qvec qj_anc);
    QCircuit T_circuitv1(qvec qi, qvec qj, qvec qj_anc);
    QCircuit T_circuitv2(qvec qi, qvec qi_anc, qvec qj, qvec qj_anc);
    QCircuit W_circuit(qvec qi, qvec qi_anc, qvec qj, qvec qj_anc);
    QCircuit Chebyshev(size_t n, qvec qi, qvec qi_anc, qvec qj, qvec qj_anc);
    QCircuit Chebyshev_minus(size_t n, qvec qi, qvec qi_anc, qvec qj, qvec qj_anc);
    QCircuit one_iteration_qcir(qvec qt,qvec qi, qvec qi_anc, qvec qj, qvec
      qj_anc);
    QSolver(size_t grid_number);
    QSolver(std::string cfg_file = "default.cfg");
    std::vector<double> m_solution;
    vector<double> get_solution() { return m_solution; };
    void run();
private:
    std::string m_cfg_file;

    //A
    std::vector<double> m_sparse_matrix;
    //b
    std::vector<double> m_residual;
    //U
    std::vector<std::vector<size_t>> m_none_zero_block;
    std::vector<std::vector<size_t>> vvj;

    double m_normal_coef;
};
```