Описание функций, типов и переменных библиотеки quantum.h

QU::complexd ~ std::complex<double>

QU::uint ~ unsigned int

QU::uint64 ~ unsigned long long

QU::complexd* QU::gen(QU::uint n, int *status = nullptr)

Генерирует нормированный вектор состояний для n кубитов. Записывает в status 0, если не возникло ошибок.

Возвращает каждому процессу указатель на QU::complexd массив – участок полученного вектора.

QU::complexd* QU::read(const char *file, int *n, int *status = nullptr)

Читает вектор состояний из файла file. Записывает в n количество кубитов, в status 0, если не возникло ошибок.

Возвращает каждому процессу указатель на QU::complexd массив – участок полученного вектора.

int QU::write(const char *file, const QU::complexd *A, QU::uint n)

Записывает вектор состояний A для n кубитов в файл file.

Возвращает 0, если не возникло ошибок.

QU::complexd* QU::transformation(const QU::complexd *A, QU::uint n, QU::uint i,

QU::complexd **P, const QU::uint *k, int *status = nullptr)

Производит i-кубитное квантовое преобразование вектора состояний A для n кубитов квантовым вентилем P (указатель на двумерный QU::complexd массив размера $2^i \times 2^i$) над кубитами k (указатель на QU::uint массив размера i). Записывает в status 0, если не возникло ошибок.

Возвращает каждому процессу указатель на QU::complexd массив – участок полученного вектора.

QU::complexd* QU::Hadamard(const QU::complexd *A, QU::uint n, QU::uint k, int *status = nullptr)

Производит однокубитное квантовое преобразование Адамара вектора состояний A для n кубитов над кубитом k. Записывает в status 0, если не возникло ошибок.

Возвращает каждому процессу указатель на QU::complexd массив – участок полученного вектора.

QU::complexd* QU::nHadamard(const QU::complexd *A, QU::uint n, int *status = nullptr)

Производит n-Адамар преобразование вектора состояний A для n кубитов. Записывает в status 0, если не возникло ошибок.

Возвращает каждому процессу указатель на QU::complexd массив – участок полученного вектора.

QU::complexd* QU::R(const QU::complexd *A, QU::uint n, QU::uint k, double alfa,

int *status = nullptr)

Производит фазовый сдвиг на угол alfa вектора состояний A для n кубитов над кубитом k. Записывает в status 0, если не возникло ошибок.

Возвращает каждому процессу указатель на QU::complexd массив – участок полученного вектора.

QU::complexd* QU::NOT(const QU::complexd *A, QU::uint n, QU::uint k, int *status = nullptr)

Производит отрицание вектора состояний A для n кубитов над кубитом k. Записывает в status 0, если не возникло ошибок.

Возвращает каждому процессу указатель на QU::complexd массив – участок полученного вектора.

QU::complexd* CNOT(const QU::complexd *A, QU::uint n, QU::uint k1, QU::uint k2,

int *status = nullptr)

Производит контролируемое отрицание вектора состояний A для n кубитов над кубитами k1 и k2. Записывает в status 0, если не возникло ошибок.

Возвращает каждому процессу указатель на QU::complexd массив – участок полученного вектора.

QU::complexd* CR(const QU::complexd *A, QU::uint n, QU::uint k1, QU::uint k2, double alfa,

int *status = nullptr)

Производит контролируемый поворот на угол alfa вектора состояний A для n кубитов над кубитами k1 и k2. Записывает в status 0, если не возникло ошибок.

Возвращает каждому процессу указатель на QU::complexd массив – участок полученного вектора.

const char* error_comment(int status)

Возвращает комментарий к ошибке с указанным статусом.

int QU::details::init(void)

Инициализирует библиотеку. Устанавливает значения переменных, указанных ниже.

Возвращает 0, если не возникло ошибок.

int QU::details::rank	номер MPI процесса
int QU::details::size	количество MPI процессов (должно быть степенью двойки)
int QU::details::log_size	двоичный логарифм от QU::details::size
int QU::details::threads	количество OpenMP потоков
bool QU::details::init_flag	true, если QU::details::init() уже вызывалась

void QU::details::get_num_threads(int *n)

Записывает в n количество OpenMP потоков, полученное из переменной окружения OMP_NUM_THREADS.

QU::uint64 QU::details::num_of_doubles(int n)

Вычисляет размер участка вектора состояний для n кубитов на процессе (размер массива).

В случае ошибки возвращает 0.

T* QU::details::get_masks<T>(QU::uint i, const QU::uint* k, QU::uint len)

Вычисляет побитовые маски, последовательно переключая биты с номерами из QU::uint массива k длины i (нумерация начиная с позиции len слева направо).

Возвращает указатель на T массив размера 2^i.

QU::uint* QU::details::get_ranks(QU::uint i, const QU::uint* k)

Вычисляет номера MPI процессов, с которыми текущему процессу необходимо совершить обмен данными, для выполнения i-кубитного квантового преобразования над кубитами k (указатель на QU::uint массив размера i).

Возвращает указатель на QU::uint массив, нулевой элемент которого – количество процессов n, остальные n элементов – номера этих процессов.

int QU::details::trans(const QU::complexd *A, QU::complexd *B, QU::uint n, QU::uint i,

QU::complexd **P, const QU::uint *k, QU::complexd *BUF = nullptr)

Производит i-кубитное квантовое преобразование вектора состояний A для n кубитов квантовым вентилем P (указатель на двумерный QU::complexd массив размера $2^i \times 2^i$) над кубитами k (указатель на QU::uint массив размера i). Записывает полученный вектор (участок для каждого процесса) в B. Позволяет для оптимизации напрямую указывать буфер BUF для хранения данных, присылаемых с других процессов (должен быть не меньше необходимого размера).

Возвращает 0, если не возникло ошибок.