Код

Задача 1

```
// 1. Надо просуммировать массив при помощи MPI.
// Каждый отдельный процесс суммирует свою часть при помощи OpenMP
void Task1(int * arr, int taskSize)
{
    int procSize, procRank;
    MPI_Comm_rank(MPI_COMM_WORLD, &procRank);
    MPI_Comm_size(MPI_COMM_WORLD, &procSize);

    bool isMain = procRank == 0;
    if (isMain)
    {
        int procSizeWithoutMaster = procSize - 1;
        int partSize = taskSize / procSizeWithoutMaster;
        for (int i = 0; i < procSizeWithoutMaster; i++)
        {
            MPI_Send(&arr[i*partSize], partSize, MPI_INT, i + 1, 0, MPI_COMM_WORLD);
            cout << "From master to " << i + 1 << " from " << i*partSize << " to " <<
i*partSize + partSize << endl;
        }

        int * partSumms = new int[procSizeWithoutMaster];
        for (int i = 0; i < procSizeWithoutMaster; i++)
        {
            MPI_Status recvStatus;
            MPI_Recv(&partSumms[i], 1, MPI_INT, i+1, MPI_ANY_TAG, MPI_COMM_WORLD,
&recvStatus);
        }

        int totalSumm = 0;
        for (int i = 0; i < procSizeWithoutMaster; i++)
            totalSumm += partSumms[i];

        delete[] partSumms;
        cout << "MPI summ(" << taskSize << ") with 1 is " << totalSumm << endl;

    }
    else
    {
        MPI_Status probeStatus;
        MPI_Probe(MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &probeStatus);

        cout << probeStatus.count;
        int size = 333;
        int *partOfArr = new int[size];
        MPI_Status recvStatus;
        MPI_Recv(partOfArr, size, MPI_INT, 0, MPI_ANY_TAG, MPI_COMM_WORLD, &recvStatus);

        int summOfPart = 0;
        for (int i = 0; i < size; i++)
            summOfPart += partOfArr[i];

        MPI_Send(&summOfPart, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
        delete[] partOfArr;
    }
}
```

Задача 2

```
/ 2. Есть главный процесс, есть подчиненные
// Подчиненные генерирует рандомные массивы рандомного типа данных и
// рандомной длины и отправляют главному
// В какой - то моомент времени рандомом
// подчиненный процесс посылает какой - нибудь стоп - код,
// выключается, а главный больше его не опрашиватъ
void Task2()
{
    int procSize, procRank;
    MPI_Comm_rank(MPI_COMM_WORLD, &procRank);
    MPI_Comm_size(MPI_COMM_WORLD, &procSize);

    bool isMain = procRank == 0;
    if (isMain)
    {
        for(int j = 0; j < 2; j++)
        for (int proc = 1; proc < procSize; proc++)
        {
            int dataType;
            MPI_Status recvStatus;
            MPI_Recv(&dataType, 1, MPI_INT, proc, MPI_ANY_TAG, MPI_COMM_WORLD, &recvStatus);


            MPI_Status probeStatus;
            MPI_Probe(proc, MPI_ANY_TAG, MPI_COMM_WORLD, &probeStatus);

            int count;
            MPI_Get_count(&probeStatus, dataType, &count);

            int maxSize = 8;
            void * data = new char[count * maxSize];
            MPI_Status dataRecvStatus;
            MPI_Recv(data, count, dataType, proc, MPI_ANY_TAG, MPI_COMM_WORLD,
&dataRecvStatus);

            cout << "Master got from " << dataRecvStatus.MPI_SOURCE << endl;

            char * charArr;
            int * intArr;
            double * doubleArr;
            long int * longIntArr;

            switch (dataType)
            {
            case MPI_CHAR:
                charArr = (char *)data;
                PrintArray(charArr, count);
                break;
            case MPI_INT:
                intArr = (int *)data;
                PrintArray(intArr, count);
                break;
            case MPI_DOUBLE:
                doubleArr = (double *)data;
                PrintArray(doubleArr, count);
                break;
            case MPI_LONG_INT:
                longIntArr = (long int *)data;
                PrintArray(longIntArr, count);
```

```
            break;
        }

        _sleep(1);
    }
}
else
{
    srand(procRank);
    int dataType;
    bool stopCode = false;
    void * data;

    double * doubleArr;
    char * charArr;
    int * intArr;
    long int * longIntArr;
    char * stopMessage;

    while (!stopCode)
    {
        int choise = rand() % 5 + 1;
        int size = rand() % 20 + 1;
        switch (choise)
        {
        case 1:
            dataType = MPI_CHAR;
            charArr = new char[size];
            for (int i = 0; i < size; i++)
                charArr[i] = 'C';
            data = charArr;
            break;
        case 2:
            dataType = MPI_DOUBLE;
            doubleArr = new double[size];
            for (int i = 0; i < size; i++)
                doubleArr[i] = 11.23;
            data = doubleArr;
            break;
        case 3:
            dataType = MPI_INT;
            intArr = new int[size];
            for (int i = 0; i < size; i++)
                intArr[i] = 43;
            data = intArr;
            break;
        case 4:
            dataType = MPI_LONG_INT;
            longIntArr = new long int[size];
            for (int i = 0; i < size; i++)
                longIntArr[i] = 12312312;
            data = longIntArr;
            break;
        case 5:
            dataType = MPI_CHAR;
            stopMessage = new char[size];
            for (int i = 0; i < size; i++)
                stopMessage[i] = 'X';
            stopCode = true;
            data = stopMessage;
            break;
```

```
            }

            MPI_Send(&dataType, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
            MPI_Send(data, size, dataType, 0, 0, MPI_COMM_WORLD);
            delete[] data;
            _sleep(1);
        }
    }
}
```

Результат

Задача 1

D:\My\Workspace\C++\MPI Practice\Debug>mpiexec.exe -n 4 "MPI Practice.exe"
Serial summ: 1000 time: 6.84214e-06
From master to 1 from 0 to 333
From master to 2 from 333 to 666
From master to 3 from 666 to 999
MPI summ(1000) with 1 is 999
MPI time is: 0.000648

Задача 2

D:\My\Workspace\C++\MPI Practice\Debug>mpiexec.exe -n 4 "MPI Practice.exe"
Master got from 1
11.23 11.23 11.23 11.23 11.23 11.23 11.23 11.23
Master got from 2
C C C C C C C C C C C C C C C C
Master got from 3
12312312 12312312 12312312 12312312 12312312 12312312 12312312 12312312 12312312
12312312 12312312 12312312 12312312 12312312 12312312 12312312 12312312
Master got from 1
X
Master got from 2
12312312 12312312 12312312 12312312 12312312 12312312 12312312 12312312 12312312
12312312 12312312 12312312 12312312 12312312 12312312 12312312
Master got from 3
X X X X X X X X X X X