Инструкция по заполнению BSF-каркаса

1. Определить в *Problem-bsfParameters.h* константные параметры модели BSF.
2. Определить в *Problem-Parameters.h* константные параметры задачи (размерность, число уравнений и т.п.).
3. Определить в *Problem-bsfTypes.h* предопределенные типы:
   - *PT_bsf_data_T*: описывает структуру данных, передаваемых каждому рабочему для выполнения очередной итерации (может содержать текущее приближение и другие параметры);
   - *PT_bsf_mapElem_T*: описывает структуру элемента списка «Map»;
   - *PT_bsf_reduceElem_T*: описывает структуру элемента списка «Reduce».
4. Определить в *Problem-Types.h* необходимые типы данных.
5. Определить переменные (Problem Variables) и структуры данных (Problem Structures) в *Problem_Data.h*.
6. Добавить в *Problem-Forwards.h* предописания пользовательских функций (User Function Forwards).
7. Добавить в *Problem-Include.h* включаемые библиотеки.
8. Реализовать в *Problem-bsfCode.cpp* предопределенные проблемно-зависимые функции (Predefined Problem-Dependent Functions):
   - *PC_bsf_AssignListSize(int* listSize)*: присваивает переменной **listSize* длину списка «Map» (совпадает с длиной списка «Reduce»);
   - *PC_bsf_CopyData(PT_bsf_data_T* dataIn, PT_bsf_data_T* dataOut)*: копирует данные из структуры **dataOut* в структуру **dataIn*;
   - *PC_bsf_IterOutput(PT_bsf_reduceElem_T* reduceResult, int count, PT_bsf_data_T data, int iterCount, double elapsedTime)*: выводит результаты итерации, используя в качестве параметров *reduceResult* – результат выполнения функции Reduce над всем списком; *count* – количество элементов, участвовавших в Reduce; *data* – задание, выполненное на данной итерации; *iterCount* – количество выполненных итераций; *elapsedTime* – общее время, затраченное на решение задачи.
   - *PC_bsf_Init(bool* success)*: выделяет память для проблемно-зависимых структур данных и выполняет их инициализацию; если необходимую память выделить не удалось, переменной **success* должно быть присвоено значение *false*;
   - *PC_bsf_MapF(PT_bsf_mapElem_T* mapElem, PT_bsf_reduceElem_T* reduceElem, PT_bsf_data_T* data, int* success)*: вычисляет функцию *F*, являющуюся параметром оператора *Map*, используя аргументы **mapElem* – элемент списка «*Ma*р», над которым выполняется функция *F*; **reduceElem* – элемент списка «*Reduce*», которому должно быть присвоено значение функции *F*; **data* – задание, содержащее текущее приближение; параметру **success* должно быть присвоено значение 0, если полученный элемент списка «*Reduce*» должен игнорироваться при выполнении операции *Reduce*.
   - *PC_bsf_ParametersOutput(int numOfWorkers, PT_bsf_data_T data)*: выводит начальные параметры задачи, используя аргументы *numOfWorkers* – количество процессов-рабочих и *data* – начальное задание, содержащее начальное приближение;

- *PC_bsf_ProblemOutput(PT_bsf_reduceElem_T* reduceResult, int count, PT_bsf_data_T data, int iterCount, double t)*: выводит конечные результаты работы программы, используя аргументы **reduceResult* – результат выполнения оператора Reduce, *count* количество элементов «суммированных» при выполнении оператора Reduce, *data* – последнее задание (последнее приближение), *t* – общее время работы программы;
- *PC_bsf_ProcessResults(bool* exit, PT_bsf_reduceElem_T* reduceResult, int count, PT_bsf_data_T* data)*: обрабатывает результаты, полученные в результате выполнения очередной итерации, используя аргументы **reduceResult* – результат выполнения оператора Reduce, *count* количество элементов «суммированных» при выполнении оператора Reduce, *data* – последнее задание (последнее приближение); если вычисления необходимо продолжить, переменной **exit* присваивается значение *true*, в противном случае – *false*;
- *PC_bsf_ReduceF(PT_bsf_reduceElem_T* x, PT_bsf_reduceElem_T* y, PT_bsf_reduceElem_T* z)*: реализует функцию, являющуюся аргументом оператора Reduce, по формуле $z := x \oplus y$.
- *PC_bsf_SetInitApproximation(PT_bsf_data_T* data)*: записывает в **data* начальное задание (начальное приближение);
- *PC_bsf_SetMapSubList(PT_bsf_mapElem_T* subList, int count, int offset, bool* success)*: заполняет подсписок списка «Map», используя аргументы **subList* – указатель на первый элемент подсписка, *count* – длина подсписка, *offset* – сдвиг, относительно начала списка; если у элемента списка «Map» имеются поля типа указатель, то необходимо выделить память под вектор, матрицу или другую структуру; если обнаружена нехватка памяти, переменной **success* необходимо присвоить значение *false*.