	Евразийский национальный университет им. Л.Н. Гумилева	Учебно-методический комплекс дисциплины	Издание: седьмое
---	--	---	------------------

### 3. Планы проведения лабораторных занятий и методические рекомендации по выполнению заданий

Методические рекомендации по выполнению лабораторных работ.

Основной упор в методике проведения лабораторных работ должен быть сделан на отработке и закреплении учебного материала в процессе выполнения заданий с применением вычислительной техники в компьютерном классе. Особое внимание при этом должно быть уделено применению элементов проблемного и контекстного обучения, опережающей самостоятельной работе студентов. Текущий контроль усвоения знаний осуществляется путем подготовки и сдачи отчетов по итогам выполнения практических работ, проверки выполнения домашнего задания, опросов на практических занятиях.

1. Ознакомьтесь с содержанием теоретической части лабораторной работы.
2. Найдите дополнительную информацию самостоятельно.
3. Выполните задания лабораторной работы
4. Подготовьте отчет для сдачи преподавателю.

Требования к оформлению отчета Отчет должен содержать:

- название и цели работы;
- результаты выполненного задания;
- общие выводы, сделанные в процессе выполнения лабораторной работы.

Защита отчета сопровождается отчетом о выполненной работе.

#### Лабораторная работа №1 Оценка трудоемкости вычислительных задач.


##### Сопоставление эффективности алгоритмов решения СЛАУ и спектрального анализа

Содержание работы: Построение параллельных алгоритмов и OpenMP-программы для численных методов прямоугольника, трапеции, Симпсона. Анализ и время выполнения программы.

Использовать в качестве вспомогательного шаблона следующий пример параллелизации:

```
c$omp parallel
  10 wrk (id) = junk (id)
  res (id) = wrk (id)**2
  if (conv (res)) goto 10
c$omp end parallel
  print *, id

c$omp parallel
c$omp& shared (var1, var2, ...)
c$omp& private (var1, var2, ...)
c$omp& firstprivate (var1, var2, ...)
c$omp& lastprivate (var1, var2, ...)
c$omp& reduction (operator | intrinsic: var1, var2, ...)
c$omp& if (expression)
c$omp& default (private | shared | none)
  [Структурный блок программы]
c$omp end parallel
```

	Евразийский национальный университет им. Л.Н. Гумилева	Учебно-методический комплекс дисциплины	Издание: седьмое
---	--	---	------------------

## Лабораторная работа №2 Повышение эффективности последовательной программы на примере операций с матрицами

Содержание работы: Построение параллельного алгоритма и OpenMP-программы для умножения матрицы на матрицу. Анализ и время выполнения программы.

Использовать в качестве вспомогательного шаблона слеующий пример параллелизации:


```
c$omp parallel
c$omp sections
c$omp section
    call computeXpart ( )
c$omp section
    call computeYpart ( )
c$omp section
    call computeZpart ( )
c$omp end sections
c$omp end parallel
    call sum ( )
```

## Лабораторная работа №3 Задание опций компилятора. Применение директивы компилятора для распараллеливания вычислений. Определение параллельного региона и параллелизация циклов

Содержание работы: Построение параллельного алгоритма и OpenMP-программы для решения системы уравнений с трехдиагональной матрицей методом прогонки. Анализ и время выполнения программы

В качестве шаблона можно использовать фрагмент программы, реализующей алгоритм прогонки:

```
x=0.
do i=0,n-1
y=h
    do j=1,n-1
a(j)=-tau/h/h*s(x,y+h/2.)
b(j)=1.+tau/h/h*(s(x,y+h/2.)+s(x,y-h/2.))
c(j)=-tau/h/h*s(x,y-h/2.)
d(j)=u1(i,j)+tau*f2(i,j)
y=y+h
    enddo
!прямая прогонка
!p(0)=0.;q(0)=u2(i,0)
p(0)=1.;q(0)=0.
do j=1,n-1
p(j)=-c(j)/(b(j)+a(j)*p(j-1))
q(j)=(d(j)-a(j)*q(j-1))/(b(j)+a(j)*p(j-1))
    enddo
!обратная прогонка
do j=n-1,0,-1
u2(i,j)=p(j)*u2(i,j+1)+q(j)
    enddo
x=x+h
enddo
k=k+1
```

	Евразийский национальный университет им. Л.Н. Гумилева	Учебно-методический комплекс дисциплины	Издание: седьмое
---	--	---	------------------

#### Лабораторная работа №4 Настройка параметров параллельного выполнения циклов при помощи ключа REDUCTION. Ключи PRIVATE, SHARED и их модификации. Область видимости переменных для параллельного региона

Содержание работы: Построение параллельного алгоритма и OpenMP-программы для решения системы линейных алгебраических уравнений методом Гаусса. Анализ и время выполнения программы.

В качестве шаблона можно использовать программу решения уравнений методом Гаусса:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#define N 6
void glavelem( int k, int a[] [N + 1], int n, int x[] );
int main()
{
    int t, a[N][N + 1], x[N]; // корни системы в виде массива + сам массив с неизвестными
    int i, j, k, n, count = 0;


    do
    {
        printf("Введите размер матрицы:\n"); // Размер системы
        scanf("%d", &n);
        if(n>N)
            printf("Слишком большое число уравнений. Повторите ввод.\n"); // Задано значение 6
        уравнений максимум (#define N 6)
        else
            printf("N = %d\n", n); // Повторный ввод системы, удовлетворяющий кол-ву уравнений
    }
    while(n>N);
    printf("Введите СЛАУ:\n"); // ввод построчно исходных данных
    for(j = 0; j < n; j++)
        for(i = 0; i < n + 1; i++)
        {
            printf("a[%d][%d] = ", j, i);
            scanf("%f", &a[j][i]);
            count++;
            if(count == (n + 1))
                printf("\n", count=0);
        }
    printf("Исходная матрица:\n"); // вывод исходных данных на экран
    for(j = 0; j < n; j++)
    {
        for(i = 0; i < n + 1; i++)
```

```
    printf("%6.3f\t",a[j][i]);
    printf("\n");
}

//прямой ход
for ( k = 0; k < n; k++ )
{ //На какой позиции должен стоять главный элемент
  glavelem( k, a, n, x ); //Установка главного элемента
  if ( fabs( a[k] [k] ) < 0.0001 )
  {
    printf( "Система не имеет единственного решения" );// Вызов функции по выбору
главного элемента
    return ( 0 );
  }
  for ( j = n; j >= k; j-- )
    a[k] [j] /= a[k] [k];
  for ( i = k + 1; i < n; i++ )
    for ( j = n; j >= k; j-- )
      a[i] [j] -= a[k] [j] * a[i] [k];

//обратный ход
for(i = 0; i < n; i++)
  x[i] = a[i][n];
for(i = n-2; i >= 0; i--)
  for(j = i+1; j < n; j++)
    x[i] = x[i] - x[j]*a[i][j]; //находим корни
//вывод матрицы квазитреугольного вида на экран
printf("Матрица приведенная к треугольному виду:\n");
for(j = 0; j < n; j++)
{
  for(i = 0; i < n + 1; i++)
    printf("%6.3f\t",a[j][i]);
  printf("\n");
}
//вывод ответа на экран
printf("Корни СЛАУ:\n");
for(i = 0; i < n; i++)
  printf("x[%d] = %6.3f\n",i,x[i]);
  getch();
}

void glavelem( int k, int a[] [N + 1], int n, int x[] ); // функция по выбору главного элемента
{
  int i, j, i_max = k, j_max = k;
  Ф ЕНУ 703-07-21 Учебно-методический комплекс дисциплины. Издание седьмое
```

	Евразийский национальный университет им. Л.Н. Гумилева	Учебно-методический комплекс дисциплины	Издание: седьмое
---	--	---	------------------

```
int temp;
```

```
for ( i = k; i < n; i++ ) //Ищем максимальный по модулю элемент
for ( j = k; j < n; j++ )
if ( fabs( a[i_max] [j_max] ) < fabs( a[i] [j] ) )
{
i_max = i;
j_max = j;
}
for ( j = k; j < n + 1; j++ ) //Переставляем строки
{
temp = a[k] [j];
a[k] [j] = a[i_max] [j];
a[i_max] [j] = temp;
}
for ( i = 0; i < n; i++ ) //Переставляем столбцы
{
temp = a[i] [k];
a[i] [k] = a[i] [j_max];
a[i] [j_max] = temp;
}
```


### **Лабораторная работа №5 Программирование явных двухслойных разностных схем с параллельной организацией вычислений. Параметры STATIC, DYNAMIC, GUIDED, RUNTIME.**

Содержание работы: Построение параллельного алгоритма и OpenMP- программы для решения системы линейных алгебраических уравнений методом простой итерации и Гаусса-Зейделя. Анализ и время выполнения программы.

В качестве шаблона можно использовать программу решения системы уравнений методом простой итерации:

```
Program MetodProstIter; {метод простых итераций}
Var
n:integer; {количество переменных или количество уравнений, как кому удобно - }
{в любом случае они должны быть равны (m=n)}
A:array of array of real; {матрица коэффициентов}
b:array Of real; {вектор-столбец свободных членов}

C:array of Array of real; {матрица Якоби - итерационная форма матрицы A}
d:array of real; {итерационная форма вектора свободных членов}
Err:Boolean; {переменная, по значению которой после выполнения процедуры
проверки}
{сходимости определяется соответствие-несоответствие условию сходимости}
X:array of Real; {вектор неизвестных}
procedure InputA(var n:integer); {ввод матрицы A}
var
```

	Евразийский национальный университет им. Л.Н. Гумилева	Учебно-методический комплекс дисциплины	Издание: седьмое
---	--	---	------------------

```

i,j:Integer;
begin
  SetLength(A,n); {именно эта встроенная процедура задает правую границу массива}
{в зависимости от количества переменных}
  for i:=0 To n-1 Do
  begin
    SetLength(A[i],n); {многомерные массивы в PascalABC можно определять как
массивы массивов}
  end;
  for i:=0 To n-1 Do
  begin
    for j:=0 To n-1 Do
    begin
      read(A[i,j]);
    end;
    writeln("");
  end;
end;

```

### Лабораторная работа №6 Алгоритм параллелизации схемы бегущего счета.

#### Директивы **ORDERED, SINGLE, FLUSH, BARRIER, ATOMIC**


Содержание работы: Построение параллельного алгоритма и OpenMP-программы для решения задачи Коши методом Эйлера, Анализ и время выполнения программы.

Алгоритм решения можно использовать следующий:

```

function F(X: Double; Y: Double): Double;
begin
  //Ваша функция
  result := Y - 2*x/Y;
end;
procedure TForm1.Euler(x, x1, y: Double; n: Integer);
var
  i: Integer;
  f1, h, y1 : Double;
begin
  h := (x1-x)/n;
  y1 := y;
  i := 1;
  repeat
    F1 := F(x, y);
    x := x+h;
    y := y+F1*h;
    y := y1+h*(F1+F(x, y))/2;
  //Вывод решения в таблицу
  ListBox1.Items.Add(FormatFloat('y(0.###) = ', x) +

```

	Евразийский национальный университет им. Л.Н. Гумилева	Учебно-методический комплекс дисциплины	Издание: седьмое
---	--	---	------------------

```

FormatFloat('0.###', y));
//Вывод решения на график
Series1.AddXY(x, y);
y1 := y;
i := i+1;
until i>n
end;
```

### **Лабораторная работа №7 Компилирование программы на кластерной системе, определение и задание параметров параллелизации**

Напишите программу, которая читает целое значение с терминала и посылает это значение всем MPI–процессам. Каждый процесс должен печатать свой номер и полученное значение. Значения следует читать, пока не появится на входе отрицательное целое число. В MPI базисной операцией отправки является операция:

```
MPI_Send (address, count, datatype, destination, tag, comm),
```

где (address, count, datatype) – количество (count) объектов типа datatype, начинающихся с адреса address в буфере отправки;

destination – номер получателя в группе, определяемой коммутатором comm;

tag – целое число, используемое для описания сообщения;

comm – идентификатор группы процессов и коммутационный контекст.

Базисной операцией приема является операция:

```
MPI_Recv (address, maxcount, datatype, source, tag, comm, status),
```

где (address, count, datatype) описывают буфер приемника, как в случае MPI\_Send;

source – номер процесса-отправителя сообщения в группе, определяемой коммутатором comm;

status – содержит информацию относительно фактического размера сообщения, источника и тэга.

Source, tag, count фактически полученного сообщения восстанавливаются на основе status.

### **Лабораторная работа №8 Программирование обмена сообщениями без блокировки**

Напишите программу, которая реализует параллельный алгоритм для задачи нахождения скалярного произведения двух векторов.


Основные элементы конструирования алгоритма и директивы следуют брать из следующих шаблонов программ:

```

call MPI_COMM_SIZE(MPI_COMM_WORLD, size, ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)
next = rank+1
if(next .eq. size) next = MPI_PROC_NULL
call MPI_SEND(buf, 1, MPI_REAL, next,
& 5, MPI_COMM_WORLD, ierr)
```

```

program lab8
include 'mpif.h'
integer BUFSIZE
```

	Евразийский национальный университет им. Л.Н. Гумилева	Учебно-методический комплекс дисциплины	Издание: седьмое
---	---	---	------------------

```

parameter (BUFSIZE = 4 + MPI_BSEND_OVERHEAD)
byte buf(BUFSIZE)
integer rank, ierr, ibufsize, rbuf
integer status(MPI_STATUS_SIZE)
call MPI_INIT(ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)
if(rank .eq. 0) then
  call MPI_BUFFER_ATTACH(buf, BUFSIZE, ierr)
  call MPI_BSEND(rank, 1, MPI_INTEGER, 1, 5,
    & MPI_COMM_WORLD, ierr)
  call MPI_BUFFER_DETACH(buf, ibufsize, ierr)
end if
if(rank .eq. 1) then
  call MPI_RECV(rbuf, 1, MPI_INTEGER, 0, 5,
    & MPI_COMM_WORLD, status, ierr)
  print *, 'Process 1 received ', rbuf, ' from process ',
    & status(MPI_SOURCE)
end if
call MPI_FINALIZE(ierr)
end

```

### Лабораторная работа №9 Программирование обмена сообщениями с использованием буфера

Напишите программу, которая реализует параллельный алгоритм для произведения вектора на матрицу. Будем считать, что матрица и вектор генерируются в нулевом процессе, затем рассылаются всем процессам. Каждый процесс считает  $n/size$  элементов результирующего вектора, где  $n$  – количество строк матрицы,  $size$  – число процессов приложения.

Основные элементы конструирования алгоритма и директивы следуют брать из следующих фрагментов программ, использующих параллелизацию:


```

program AutoInverse1 !program for change inclision center
integer i,m,n,nz,k_pol,k_point0,k_point,ns,IterMax,InnerIterMax
  include "mpif.h" !! This brings in pre-defined MPI constants, ...
  integer Iam, p,tag, ierr, status_(MPI_STATUS_SIZE),req(1)
real*8,parameter :: Pi=3.14159265358979
real*8 beta,xp,yp,zmax,smin,smax,ro1,ro2,JdevTotal0,J_t,eps,epsJ
real*8 rd_pixel(0:129),phi(0:129),a_pol(0:64),J_pol(0:64)
real*8 rd_pixel0(0:129),phi0(0:129)
real*8 teta(0:128),z(0:128)
real*8 r_app(0:128,0:64),r_app0(0:128,0:64)

common/b/ rd_pixel,phi,rd_pixel0,phi0
common/c/ teta,z,a_pol
common/d/ r_app,r_app0
common/e/ tag, ierr, status_,req
  call MPI_Init(ierr) !! starts MPI
  call MPI_Comm_rank(MPI_COMM_WORLD, Iam, ierr) !! get current proc id
  call MPI_Comm_size(MPI_COMM_WORLD, p, ierr) !! get number of procs

```



	Евразийский национальный университет им. Л.Н. Гумилева	Учебно-методический комплекс дисциплины	Издание: седьмое
---	--	---	------------------

```

tag=123
call Loaddata() !measures
call MPI_BARRIER(MPI_COMM_WORLD,ierr)
call Jdev(k_pol,ns,J_pol,J_t,Iam,p) !initial error

m=32
n=m*2 !polar point number
nz=32 !points by z
k_point=3 !point for approx contour
call SetInitialContour(k_point0,k_point)
call SetMesh(m,nz,zmax,teta,z)
!write(*,*) 'teta,z meshes'
!write(*,*) (teta(i),i=0,m*2)
!write(*,*) (z(i),i=0,m*2)
if (Iam==0) then
  OPEN(12,FILE='./Test/his4MPI.txt')
  call History(xp,yp,rd_pixel,phi,0)!initial guess
endif
IterMax=50 ; InnerIterMax=20
eps=1.D-07; epsJ=0.1D+0
call AutoInverse(IterMax,k_point,m,nz,InnerIterMax,xp,yp,eps,epsJ,Iam,p) ;

if (Iam==0) then
  CLOSE(12)
  write(*,*)'I finished!'
endif
call MPI_Finalize(ierr)

```

### Лабораторная работа №10 Использование встроенных функций MPI для измерения временных характеристик системы

Напишите программу для измерения времени, необходимого для выполнения MPI\_Allreduce на MPI\_COMM\_WORLD. Как изменяются характеристики для MPI\_Allreduce при изменении размера MPI\_COMM\_WORLD?


В следующую работающую программу требуется вставить процедуры измерения времени и MPI\_Allreduce

```

implicit none
integer n, p, i, j, k, ierr, master
real h, a, b, integral, pi
integer req(1)
include "mpif.h" !! This brings in pre-defined MPI constants, ...
integer Iam, source, dest, tag, status(MPI_STATUS_SIZE)
real my_result, Total_result, result
data master/0/
c**Starts MPI processes ...
  call MPI_Init(ierr) !! starts MPI
  call MPI_Comm_rank(MPI_COMM_WORLD, Iam, ierr)!! get current proc id
  call MPI_Comm_size(MPI_COMM_WORLD, p, ierr)!! get number of procs

  pi = acos(-1.0)    !! = 3.14159...
  a = 0.0            !! lower limit of integration
  b = pi/2.          !! upper limit of integration
  n = 500            !! number of increment within each process

```

	Евразийский национальный университет им. Л.Н. Гумилева	Учебно-методический комплекс дисциплины	Издание: седьмое
---	--	---	------------------

```

dest = master      !! define the process that computes the final result
tag = 123          !! set the tag to identify this particular job
h = (b-a)/n/p      !! length of increment
my_result = integral(a,Iam,h,n)
write(*,*) 'Iam=',Iam,', my_result=',my_result
if(Iam.eq.master) then      ! the following is serial
    result = my_result
    do k=1,p-1
        call MPI_Recv(my_result, 1, MPI_REAL,
& MPI_ANY_SOURCE, tag,      !! more efficient, less prone to
deadlock
& MPI_COMM_WORLD, status, ierr) !! root receives my_result from
proc
        result = result + my_result
    enddo
else
    call MPI_Isend(my_result, 1, MPI_REAL, dest, tag,
& MPI_COMM_WORLD, req, ierr)    !! send my_result to intended
dest.
    call MPI_Wait(req, status, ierr) !! wait for nonblock send ...
endif


c**results from all procs have been collected and summed ...
if(Iam.eq. 0) then
    write(*,*) 'Final Result =',result
endif
call MPI_Finalize(ierr)      !! let MPI finish up ...
stop
end
real function integral(a,i,h,n)
implicit none
integer n, i, j
real h, h2, aij, a
real fct, x
fct(x) = cos(x)              !! kernel of the integral
integral = 0.0               !! initialize integral
h2 = h/2.
do j=0,n-1                   !! sum over all "j" integrals
    aij = a + (i*n+j)*h      !! lower limit of "j" integral
    integral = integral + fct(aij+h2)*h
enddo
return
end

```

### Лабораторная работа №11 Программирование обмена сообщениями с использованием блокировки

Напишите программу для измерения времени передачи вещественных данных двойной точности от одного процесса другому. Выполните задание при условии, что каждый процесс передает и принимает от процесса, находящегося на расстоянии  $\text{size}/2$ , где имеется  $\text{size}$  процессов в `MPI_COMM_WORLD`. Лучшее решение будет получено при использовании `MPI_SendRecv`, `MPI_Barrier`, чтобы гарантировать, что различные пары стартуют почти одновременно, однако возможны другие решения. Для усреднения накладных расходов следует: повторить остаточное количество операций пересылок для получения времени в пределах олей секунды (образцовое решение делает  $100000/\text{size}$  итераций для целых  $\text{size}$ ), повторить тестирование несколько раз (например, 10) и усреднить результаты.

Использовать предыдущую программу с заменой операций обмена на операции блокировкой

	Евразийский национальный университет им. Л.Н. Гумилева	Учебно-методический комплекс дисциплины	Издание: седьмое
---	--	---	------------------

## Лабораторная работа №12 Программирование обмена сообщениями с измерением временных параметров

Напишите программу для измерения времени, необходимого для выполнения MPI\_Allreduce на MPI\_COMM\_WORLD. Как изменяются характеристики для MPI\_Allreduce при изменении размера MPI\_COMM\_WORLD?

Напишите программу для измерения времени, необходимого для выполнения MPI\_Barrier на MPI\_COMM\_WORLD. Как изменяются характеристики для MPI\_Barrier при изменении размера MPI\_COMM\_WORLD?

program example12

```
include 'mpif.h'
```

```
integer ierr, rank, size, prev, next, reqs(4), buf(2)
```

```
integer stats(MPI_STATUS_SIZE, 4)
```

```
call MPI_INIT(ierr)
```

```
call MPI_COMM_SIZE(MPI_COMM_WORLD, size, ierr)
```

```
call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)
```

```
prev = rank - 1
```

```
next = rank + 1
```

```
if (rank .eq. 0) prev = size - 1
```

```
if (rank .eq. size - 1) next = 0
```

```
call MPI_Irecv(buf(1), 1, MPI_INTEGER, prev, 5,  
& MPI_COMM_WORLD, reqs(1), ierr)
```

```
call MPI_Irecv(buf(2), 1, MPI_INTEGER, next, 6,  
& MPI_COMM_WORLD, reqs(2), ierr)
```

```
call MPI_Isend(rank, 1, MPI_INTEGER, prev, 6,  
& MPI_COMM_WORLD, reqs(3), ierr)
```

```
call MPI_Isend(rank, 1, MPI_INTEGER, next, 5,  
& MPI_COMM_WORLD, reqs(4), ierr)
```

```
call MPI_WAITALL(4, reqs, stats, ierr);
```

```
print *, 'process ', rank,
```

```
& ' prev=', buf(1), ' next=', buf(2)
```

```
call MPI_
```

## Лабораторная работа №13 Организация асинхронной передачи данных. Обнаружение тупиковых ситуаций и разрешение

Напишите программу для определения объема буферизации, необходимого для выполнения MPI\_Send. Это означает, что нужно написать программу, которая определяет, насколько большого объема сообщение может быть послано без включения соответствующего приема в процессе назначения.

Использовать в качестве шаблона следующую программу параллелизации:

program lab13

```
include 'mpif.h'
```

```
integer ierr, rank, size, prev, next, buf(2)
```

```
integer status1(MPI_STATUS_SIZE), status2(MPI_STATUS_SIZE)
```

```
call MPI_INIT(ierr)
```


```
call MPI_COMM_SIZE(MPI_COMM_WORLD, size, ierr)
```

```
call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)
```

```
prev = rank - 1
```

```
next = rank + 1
```

Ф ЕНУ 703-07-21 Учебно-методический комплекс дисциплины. Издание седьмое

	Евразийский национальный университет им. Л.Н. Гумилева	Учебно-методический комплекс дисциплины	Издание: седьмое
---	---	---	------------------

```

if(rank .eq. 0) prev = size - 1
if(rank .eq. size - 1) next = 0
call MPI_SENDRECV(rank, 1, MPI_INTEGER, prev, 6,
& buf(2), 1, MPI_INTEGER, next, 6,
& MPI_COMM_WORLD, status2, ierr)
call MPI_SENDRECV(rank, 1, MPI_INTEGER, next, 5,
& buf(1), 1, MPI_INTEGER, prev, 5,
& MPI_COMM_WORLD, status1, ierr)
print *, 'process ', rank,
& ' prev=', buf(1), ' next=', buf(2)
call MPI_FINALIZE(ierr)
end

```

#### **Лабораторная работа №14 Перезасылка разнотипных данных. Упаковка данных**


Напишите программу нахождения максимального значения и его индекс из массива чисел равномерно распределенного по процессам.

Использовать в качестве шаблона следующую программу параллелизации:

```

program example13
include 'mpif.h'
integer ierr, rank, size, MAXPROC, NTIMES, i, it
parameter (MAXPROC = 128, NTIMES = 10000)
integer ibuf(MAXPROC)
double precision time_start, time_finish
integer req(2*MAXPROC), statuses(MPI_STATUS_SIZE, MAXPROC)
call MPI_INIT(ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD, size, ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)
if(rank .eq. 0) then
do i = 1, size-1
call MPI_RECV_INIT(ibuf(i), 0, MPI_INTEGER, i, 5,
& MPI_COMM_WORLD, req(i), ierr)
call MPI_SEND_INIT(rank, 0, MPI_INTEGER, i, 6,
& MPI_COMM_WORLD, req(size+i),
& ierr)
end do
time_start = MPI_WTIME(ierr)
do it = 1, NTIMES
call MPI_STARTALL(size-1, req, ierr)
call MPI_WAITALL(size-1, req, statuses, ierr)
call MPI_STARTALL(size-1, req(size+1), ierr)
call MPI_WAITALL(size-1, req(size+1), statuses,
& ierr)
end do
else
call MPI_RECV_INIT(ibuf(1), 0, MPI_INTEGER, 0, 6,
& MPI_COMM_WORLD, req(1), ierr)
call MPI_SEND_INIT(rank, 0, MPI_INTEGER, 0, 5,
& MPI_COMM_WORLD, req(2), ierr)
time_start = MPI_WTIME(ierr)
do it = 1, NTIMES
call MPI_START(req(2), ierr)

```

	Евразийский национальный университет им. Л.Н. Гумилева	Учебно-методический комплекс дисциплины	Издание: седьмое
---	---	---	------------------

```

call MPI_WAIT(req(2), statuses, ierr)
call MPI_START(req(1), ierr)
call MPI_WAIT(req(1), statuses, ierr)
end do
end if
time_finish = MPI_WTIME(ierr)-time_start
print *, 'rank = ', rank, ' all time = ',
& (time_finish)/NTIMES
time_start = MPI_WTIME(ierr)
do it = 1, NTIMES
call MPI_BARRIER(MPI_COMM_WORLD,ierr)
enddo
time_finish = MPI_WTIME(ierr)-time_start
print *, 'rank = ', rank, ' barrier time = ',
& (time_finish)/NTIMES
call MPI_FINALIZE(ierr)
end

```