Министерство образования
Российской Федерации

Оренбургский государственный университет

В.И. Чепасов,
А.П. Тетерин, Д.А. Харченко

Алгоритмическая и программная реализация вычисления числа пи в длинной арифметике без распараллеливания и с распараллеливанием процедуры суммирования

Оренбург-2009

ББК 30.121: 22.172
Ч- 44
УДК 539.3/.6.001.24:519.2

Рецензент:
заслуженный деятель науки, д.т.н., профессор Шевеленко В.Д.

Ч- 44 Чепасов В.И.
Алгоритмическая и программная реализация вычисления числа пи в длинной арифметике без распараллеливания и с распараллеливанием процедуры суммирования – Оренбург: ОГУ, 2009. - 120с.

2402000000
Ч ———————— ББК 30.121: 22.172

СОДЕРЖАНИЕ

Стр.

ВВЕДЕНИЕ 5

Глава 1 6
Реализация вычисления числа пи в длинной арифметике

1.1 Операция сложения(язык Фортран) 6

1.2 Операция вычитания(язык Фортран) 8

1.3 Операция умножения(язык Фортран) 9

1.4 Операция деления(язык Фортран) 11

1.5 Программная реализация вычисления числа пи (язык Фортран) 12

1.6 Входные и выходные данные программ(язык Фортран) 13

1.7 Поставка программного комплекса(язык Фортран) 15

1.8 Программная реализация вычисления числа пи в «С» 16

1.9 Входные и выходные данные программ(язык С) 19

1.10 Поставка программного комплекса 19

Глава 2 20
Алгоритмическое описание
программного комплекса (язык Фортран)

2.1 Подпрограмма по реализации сложения 20
без учета знаков слагаемых

2.2 Подпрограмма по реализации сложения 22
с учетом знаков слагаемых

2.3 Подпрограмма по реализации вычитания 23
без учета знаков уменьшаемого и вычитаемого

2.4 Подпрограмма по реализации вычитания 26
с учетом знаков уменьшаемого и вычитаемого

2.5 Подпрограмма по реализации умножения без учета знаков сомножителей 26

2.6 Подпрограмма умножения с учетом знаков сомножителей 29

2.7 Подпрограмма по реализации деления без учета знаков делимого и делителя 29

2.8 Подпрограмма по реализации деления с учетом знаков делимого и делителя 31

2.9 Главная программа chispi.for,pikontr.for 32

2.10 Главная программа pidlpar1.for 32

Список литературы 33

ПРИЛОЖЕНИЕ-1(наборы с исходными текстами программ на Фортране:chispi-34c,pikontr-37c,pidlpar1-40c,sl-44c,slzn-47c,wich-50c,wiczn-54c,ymnzn-57c,delzn-63c.) 34

ПРИЛОЖЕНИЕ-2(наборы с исходными текстами программ на C: pidlpar-68c,sl-74c,slzn-77c,wich-80c,wiczn-83c,ymnzn-86c, delzn-90c) 68

ПРИЛОЖЕНИЕ-3(набор pidlmpi.c для реализации распараллеливания процедуры суммирования) 95

ПРИЛОЖЕНИЕ-4(выходные файлы с результатами) 119

Введение

На сегодняшний день все современные ЭВМ имеют конечную разрядную сетку.

В связи с этим невозможно осуществлять вычисления с необходимой степенью точности.

Поэтому был разработан алгоритм и осуществлена его программная реализация для выполнения всех арифметических операций с произвольным количеством разрядов в целой и дробной частях операндов.

На базе длинной арифметики было реализовано вычисление числа пи на однопроцессорном и многопроцессорном комплексах.

Программная реализация была сделана на алгоритмических языках Фортран,С для однопроцессорного комплекса.

Для многопроцессорного комплекса программная реализация была сделана на алгоритмическом языке С.

Глава-1 Реализация вычисления числа пи в длинной арифметике

Для реализации вычисления числа **пи** в длинной арифметике все арифметические операции сложение, вычитание, деление, умножение были реализованы в длинной арифметике для вещественных значений.

Для этого были организованы для каждого числа два массива: массив, каждый элемент которого содержал разряд целой части числа, массив, каждый элемент которого содержал разряд дробной части числа.

Кроме этого для каждого числа вводилась целая переменная, которая содержала знак этого числа.

Значение 1 соответствует положительному значению, значение (-1) отрицательному значению.

1.1 Операция сложения(язык Фортран)

Непосредственное сложение двух вещественных значений осуществляет подпрограмма sl.for (язык Фортран,приложение-1).

Сложение двух вещественных значений с учетом знаков осуществляет подпрограмма slzn.for.

В подпрограмме slzn осуществляется вызов подпрограммы sl.for для непосредственного сложения.

Вызов подпрограммы slzn.for:

call slzn (iz1, iz2, izrez, nc1, nd1, ic1, id1, nc2, nd2, ic2, id2, kcrez, kdrez, icrez, idrez,krazr)

Параметры при вызове подпрограммы slzn:

iz1 – знак первого слагаемого, целая четырехбайтовая переменная;

iz2 – знак второго слагаемого, целая четырехбайтовая переменная;

izrez – знак вычисленной суммы, целая четырехбайтовая переменная;

nc1 – количество разрядов в целой части первого слагаемого, целая четырехбайтовая переменная;

nd1 – количество разрядов в дробной части первого слагаемого, целая четырехбайтовая переменная;

ic1 – массив значений разрядов в целой части первого слагаемого, целые четырехбайтовые;

id1 – массив значений разрядов в дробной части первого слагаемого, целые четырехбайтовые;

nc2 – количество разрядов в целой части второго слагаемого, целая четырехбайтовая;

nd2 - количество разрядов в дробной части второго слагаемого, целая четырехбайтовая;

ic2 - массив значений разрядов в целой части второго слагаемого, целые четырехбайтовые;

id2 - массив значений разрядов в дробной части второго слагаемого, целые четырехбайтовые;

kcrez – количество разрядов в целой части суммы, целое четырехбайтовое;

kdrez - количество разрядов в дробной части суммы, целое четырехбайтовое;

icrez - массив значений разрядов в целой части суммы, целые четырехбайтовые;

idrez - массив значений разрядов в дробной части суммы, целые четырехбайтовые;

krazr- количество разрядов в дробной части длинной арифметики, целое четырехбайтовое.

Пользователю для реализации длинного сложения надо вызвать подпрограмму slzn.

1.2 Операция вычитания(язык Фортран)

Вычитание двух вещественных с учетом знаков осуществляет подпрограмма wiczn.for(приложение-1).

В подпрограмме wiczn осуществляется вызов подпрограммы wich для непосредственного вычитания.

Вызов подпрограммы wiczn:

call wiczn (iz1, iz2, izrez, nc1, nd1, ic1, id1, nc2, nd2, ic2, id2, kcrez, kdrez, icrez, idrez,krazr),

iz1 - знак уменьшаемого, целая четырехбайтовая,

iz2 – знак вычитаемого, целая четырехбайтовая,

izrez – знак разности, целая четырехбайтовая,

nc1 – количество разрядов в целой части уменьшаемого, целая четырехбайтовая;

nd1 – количество разрядов в дробной части уменьшаемого, целая четырехбайтовая;

ic1 – массив значений разрядов в целой части уменьшаемого, целые четырехбайтовые;

id1 - массив значений разрядов в дробной части уменьшаемого, целые четырехбайтовые;

nc2 – количество разрядов в целой части вычитаемого, целое четырехбайтовое;

nd2 - количество разрядов в дробной части вычитаемого, целое четырехбайтовое;

ic2 – массив значений разрядов в целой части вычитаемого, целые четырехбайтовые;

id2 - массив значений разрядов в дробной части вычитаемого, целые четырехбайтовые;

kcrez - количество разрядов в целой части разности, целое четырехбайтовое;

kdrez - количество разрядов в дробной части разности, целое четырехбайтовое;

icrez - массив значений разрядов в целой части разности, целые четырехбайтовые;

idrez - массив значений разрядов в дробной части разности, целые четырехбайтовые;

krazr-количество разрядов в дробной части длинной арифметики, целое четырехбайтовое.

Для реализации длинного вычитания пользователю необходимо вызвать подпрограмму wiczn.

1.3 Операция умножения(язык Фортран)

Умножение двух вещественных в длинном формате осуществляет подпрограмм ymnzn.for (с учетом знаков)(приложение-1).

В подпрограмме ymnzn осуществляется вызов подпрограммы ymn для непосредственного умножения в длинном формате.

Вызов подпрограммы ymnzn:

call ymnzn (iz1, iz2, izrez, nc1, nd1, ic1, id1, nc2, nd2, ic2, id2, kcrez, kdrez, icrez, idrez,krazr),

iz1 - знак первого сомножителя, целое четырехбайтовое;

iz2 – знак второго сомножителя, целое четырехбайтовое;

izrez – знак произведения, целое четырехбайтовое;

nc1 – количество разрядов в целой части первого сомножителя, целое четырехбайтовое;

nd1 – количество разрядов в дробной части первого сомножителя, целое четырехбайтовое;

ic1 – массив значений разрядов в целой части первого сомножителя, целые четырехбайтовые;

id1 - массив значений разрядов в дробной части первого сомножителя, целые четырехбайтовые;

nc2 – количество разрядов в целой части второго сомножителя, целое четырехбайтовое;

nd2 - количество разрядов в дробной части второго сомножителя, целое четырехбайтовое;

ic2 – массив значений разрядов в целой части второго сомножителя, целые четырехбайтовые;

id2 - массив значений разрядов в дробной части второго сомножителя, целые четырехбайтовые;

kcrez - количество разрядов в целой части произведения, целое четырехбайтовое;

kdrez - количество разрядов в дробной части произведения, целое четырехбайтовое;

icrez - массив значений разрядов в целой части произведения, целые четырехбайтовые;

idrez - массив значений разрядов в дробной части произведения, целые четырехбайтовые;

krazr-количество разрядов в дробной части длинной арифметики, целое четырехбайтовое.

Для реализации длинного умножения пользователю надо вызвать подпрограмму ymnzn.

1.4 Операция деления(язык Фортран)

Длинное деление осуществляет подпрограмма delzn.for(приложение-1) с учетом знаков вещественных значений.

Непосредственное деление осуществляет подпрограмма deld, которая вызывается в подпрограмме delzn.

Вызов подпрограммы delzn:

call delzn (iz1, iz2, izrez, nc1, nd1, ic1, id1, nc2, nd2, ic2, id2, kcrez, kdrez, icrez, idrez,krazr),

iz1 - знак делимого, целое четырехбайтовое;

iz2 – знак делителя, целое четырехбайтовое;

izrez – знак частного, целое четырехбайтовое;

nc1 – количество разрядов в целой части делимого, целое четырехбайтовое;

nd1 – количество разрядов в дробной части делимого, целое четырехбайтовое;

ic1 – массив значений разрядов в целой части делимого, целые четырехбайтовые;

id1 - массив значений разрядов в дробной части делимого, целые четырехбайтовые;

nc2 – количество разрядов в целой части делителя, целое четырехбайтовое;

nd2 - количество разрядов в дробной части делителя, целое четырехбайтовое;

ic2 – массив значений разрядов в целой части делителя, целые четырехбайтовые;

id2 - массив значений разрядов в дробной части делителя, целые четырехбайтовые;

kcrez - количество разрядов в целой части частного, целое четырехбайтовое;

kdrez - количество разрядов в дробной части частного, целое четырехбайтовое;

icrez - массив значений разрядов в целой части частного, целые четырехбайтовые;

idrez - массив значений разрядов в дробной части частного, целые четырехбайтовые;

krazr-количество разрядов в дробной части длинной арифметики, целое четырехбайтовое.

Для реализации длинного деления пользователю надо вызвать подпрограмму delzn.

1.5 Программная реализация вычисления числа пи(язык Фортран)

С помощью подпрограмм slzn, wiczn,sl, wich, ymnzn, ymn, delzn, deld реализовано вычисление числа пи.

За алгоритмическую основу взято приближенное вычисление определенного интеграла от функции y=1/(1+x*x) на отрезке [0,1].

Определенный интеграл вычислялся по формуле прямоугольников.

Главная программа по реализации вычисления числа пи в длинной арифметике chispi.for представлена в приложении-1.

В главной программе chispi.for осуществляется ввод количества разрядов в дробной части шага интегрирования, количества разрядов в дробной части длинной арифметики, вызов подпрограмм slzn, wiczn,ymnzn, delzn.

Поскольку при маленьком шаге интегрирования процесс вычисления числа пи может идти достаточно долго на однопроцессорном комплек-

се(часы,сутки,недели,месяцы и т.д.),то была написана программа pikontr.for(приложение-1) по вычислению числа пи в длинной арифметике с созданием контрольных точек.

Для функционирования программы pikontr.for необходимо в текстовом редакторе создать файл kontrpi с исходными данными(смотри пункт 1.6).

Главная программа по имитации распараллеливания в длинной арифметике при вычислении интегральной суммы pidlpar1.for(приложение-1).

В программе pidlpar1.for осуществляется ввод количества разрядов в дробной части шага интегрирования, количества разрядов в дробной части длинной арифметики,количества процессов, вызов подпрограмм slzn, wiczn,ymnzn, delzn.

1.6 Входные и выходные данные программ(язык Фортран)

В главной программе chispi.for происходит ввод исходных данных с клавиатуры:

-количество разрядов в дробной части шага интегрирования dx,

-количество разрядов в дробной части длинной арифметики.

Результаты вычисления записываются в файл ch_pi.

В главной программе pikontr.for происходит ввод исходных данных из файла kontrpi,который имеет следующую структуру:

-в первой строчке через пробел идут значения количества разрядов в дробной части шага интегрирования,количества разрядов в дробной части длинной арифметики,шаг создания контрольной точки(10000,20000 и т.д.),

-во второй строчке через пробел идут три единицы,

-в третьей строчке ноль,

-в четвертой строчке ноль,

-в пятой строчке через пробел идут три единицы,

-в шестой строчке ноль,

-в седьмой строчке ноль.

Например, файл kontrpi:

5 20 10000

1 1 1

0

0

1 1 1

0

0

В приведенном файле kontrpi в первой строчке:

5-количество разрядов в дробной части шага интегрирования,

20- количество разрядов в дробной части длинной арифметики.

10000-шаг создания контрольной точки.

Во второй строчке идут через пробел три единицы.

В третьей строчке ноль,в четвертой строчке ноль.

В пятой строчке через пробел идут три единицы,в шестой строчке ноль,

в седьмой строчке ноль.

Во время выполнения программы pikontr.for при создании одной контрольной точки будут выдаваться на экран две строчки:

создается контрольная точка,снимать нельзя!
снимать можно!

После выдачи последней строчки программу можно снять,а потом запустить для продолжения вычисления с созданной контрольной точки.

При этом пользователь для продолжения не должен ничего вводить с клавиатуры и не должен создавать никаких файлов(файл kontrpi создается один раз в начале вычисления!).

При количестве разрядов в дробной части шага интегрирования 5 ,при количестве разрядов в дробной части длинной арифметики 20,шаге контрольной точки 10000 промежуток между созданиями контрольных точек будет 40 секунд.

То есть в течении 40 секунд после выдачи строчки «**снимать можно!**» **пользователь может снять программу.**

После завершения работы программы pikontr.for результаты вычислений будут в файле ch_pi.

Для организации нового вычисления с другими исходными данными необходимо модифицировать файл kontrpi(смотри выше!).

В этот файл пишутся данные по создаваемой контрольной точке в процессе вычисления программой pikontr.for.

В главной программе pidlpar1.for происходит ввод исходных данных с клавиатуры:

-количество разрядов в дробной части шага интегрирования dx,

-количество разрядов в дробной части длинной арифметики,

-количество процессов при имитации параллельного суммирования.

Результаты вычисления записываются в файл pipardl.

1.7 Поставка программного комплекса(язык Фортран)

В поставку входят исходные модули всех подпрограмм и исходный модуль главной программы:

delzn.for – длинное деление с учетом знака. В delzn идет подпрограмма deld по непосредственному делению.

slzn.for – длинное сложение с учетом знака.

sl.for – непосредственное длинное сложение.

wiczn.for – длинное вычитание с учетом знака.

wich.for – непосредственное длинное вычитание.

ymnzn.for – длинное умножение с учетом знака. В ymnzn идет подпрограмма ymn по непосредственному длинному умножению.

chispi.for-главная программа без имитации распараллеливания суммирования.

pikontr.for- главная программа без имитации распараллеливания суммирования с созданием контрольных точек.

pidlpar1.for- главная программа с имитацией распараллеливания суммирования.

Вместе с исходными текстами программ идет поставка библиотеки под названием bibl и ее оглавления под названием obibl с объектными модулями программ.

- delzn.obj,
- slzn.obj,
- sl.obj,
- wiczn.obj,
- wich.obj,
- ymnzn.obj,

В поставку входит архив pidlpar.arj, в котором есть еще загрузочные модули chispi.exe,pikontr.exe,pidlpar1.exe,a.exe(реализация в C,смотри 1.8)

1.8 Программная реализация вычисления числа пи в «C»

В случае реализации в «C» в MS/DOS имеем следующие внешние вызываемые функции(приложение-2):

delzn,deld,sl,slzn,wich,wiczn,ymnzn, ymn.

Главная функция pidlpar.c,в которой идет имитация распараллеливания процедуры суммирования(приложение-2).

Главная функция pidlmpi.c для реализации распараллеливания процедуры суммирования на многопроцессорном комплексе (приложение-3).

Функции delzn,deld, sl,slzn, wich, wiczn, ymnzn, ymn полностью аналогичны подпрограммам под теми же именами в Фортране в случае однопроцессорной реализации.

Список параметров внешних вызываемых функций delzn,deld,sl, slzn, wich, wiczn,ymnzn,ymn полностью аналогичен списку параметров соответствующих подпрограмм в Фортране.

Но передача выходных параметров знак результата арифметической операции, количество разрядов в целой части результата, количество разрядов в дробной части результата арифметической операции идет по указателям.

Все параметры-целые двухбайтовые переменные.

Прототипы внешних вызываемых функций по реализации длинной арифметики delzn,deld, slzn, sl, wiczn, wich,ymnzn,ymn:

```
void delzn(int za,int zb,int *pzc,
int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
int *pkcdel,int *pkddel,int *cdel,int *ddel,int krazr);

void deld(int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
int *pkcdel,int *pkddel,int *cdel,int *ddel,int krazr);

void slzn(int za,int zb,int *pzc,
int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr);

void sl(int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr);
```

```
void wiczn(int za,int zb,int *pzc,
int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr);

void wich(int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr);

void ymnzn(int za,int zb,int *pzc,int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
int *pkcel,int *pkdro,int *ctot,int *cdtot,int krazr);

void ymn(int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
int *pkcel,int *pkdro,int *ctot,int *cdtot,int krazr);
```

Во внешних вызываемых функциях delzn,deld:

pzc-указатель для знака частного,

pkcdel-указатель для количества разрядов в целой части частного,

pkddel-указатель для количества разрядов в дробной части частного.

Во внешних вызываемых функциях slzn, sl, wiczn, wich,ymnzn,ymn:

pzc-указатель для знака результата суммирования,вычитания,умножения,

pkcel-указатель для количества разрядов в целой части результата суммирования,вычитания,умножения,

pkdro-указатель для количества разрядов в дробной части результата суммирования,вычитания,умножения.

Остальные параметры во внешних вызываемых функциях полностью аналогичны параметрам в соответствующих подпрограммах языка Фортран.

Элементы всех массивов, все переменные-целые двухбайтовые.

Получение загрузочного модуля идет через проект.

Проект составлялся из объектных модулей внешних вызываемых функций.

Трансляция исходных модулей происходила без проверки переполнения стека с типом памяти HUGE.

Это режим OPTIONS в главном меню,далее FULL MENU и CODE GENERATION.

Перечисленные режимы трансляции выбирали в CODE GENERATION.

1.9 Входные и выходные данные программ(язык С)

Для функционирования программы pidlpar.c(загрузочный модуль a.exe)по реализации вычисления числа пи в длинной арифметике с имитацией распараллеливания процедуры суммирования необходимо с клавиатуры осуществить ввод следующих данных:

-количество разрядов в дробной части шага интегрирования dx,

-количество разрядов в дробной части длинной арифметики,

-количество процессов при имитации параллельного суммирования.

Результаты вычисления записываются в файл pidlp_c.

1.10 Поставка программного комплекса

В поставку программ по реализации вычисления числа пи в длинной арифметике без распараллеливания и с распараллеливанием процедуры суммирования входят:

1.pidlpar.arj- архив по реализации в фортране,С для однопроцессорного и многопроцессорных комплексов

2.opdlpi.arj-архив описания программ в Фортране,С

3.totpimpi.arj-архив программы по реализации распараллеливания процедуры суммирования.

Глава -2 Алгоритмическое описание программ (язык Фортран)

2.1Подпрограмма по реализации сложения без учета знаков слагаемых

Вызов подпрограммы(приложение-1):

call sl (n, nd, a, ad, m, md, b, bd, kcel, kdro, rezc, rezdr, krazr),

n – количество разрядов в целой части первого слагаемого,
nd – количество разрядов в дробной части первого слагаемого,
a – массив разрядов целой части первого слагаемого,
ad - массив разрядов дробной части первого слагаемого,
m - количество разрядов в целой части второго слагаемого,
md - количество разрядов в дробной части второго слагаемого,
b – массив разрядов целой части второго слагаемого,
bd – массив разрядов дробной части второго слагаемого,
kcel – количество разрядов в целой части суммы,
kdro - количество разрядов в дробной части суммы,
rezc – массив разрядов целой части суммы,
rezdr - массив разрядов дробной части суммы,
krazr - количество разрядов в дробной части длинной арифметики.

В цикле do 66875 происходит обнуление массивов целой и дробной части суммы. В операторе if(n.ge.m) goto 897 идет сравнение длин целых частей слагаемых.

Если длина целой части первого слагаемого меньше длины целой части второго слагаемого, то начиная с оператора do 678 и по метку 897 идет пересылка целых и дробных частей слагаемых.

После метки 897 идет нахождение максимальной длины дробной части слагаемых, расширение в цикле do 79 дробной части второго слагаемого нулями до максимальной длины дробной части, расширение в цикле do 82 дробной части первого слагаемого нулями до максимальной длины дробной части.

После метки 81 в цикле do 100 идет формирование единого массива разрядов a(i) для первого слагаемого(разряды целой части + разряды дробной части).

В цикле do 101 идет формирование единого массива разрядов b(i) для второго слагаемого(разряды целой части + разряды дробной части).

Далее идет непосредственное поразрядное сложение.

При сложении:

i – счетчик разрядов в сумме,

p – значение переноса из младшего разряда в старший,

nt – текущий номер разряда в первом слагаемом, начинается с младшего разряда,

mt – текущий номер разряда во втором слагаемом, начинается с младшего разряда.

После метки 7 идет поразрядное суммирование элементов массивов в двух слагаемых.

В переменной i1 будет значение переноса из младшего разряда в следующий старший.

В переменной OS - результат суммирования разрядов.

Этот результат суммирования помещается в элемент c(i) массива результата суммирования.

После уменьшения номеров разрядов идет анализ текущих номеров разрядов слагаемых и значения переноса P.

После метки 67:

ik – номер младшего разряда в целой части суммы,

kcel – количество разрядов в целой части суммы,

kdro – количество разрядов в дробной части суммы.

В цикле do 1121 идет пересылка в массив rezc целой части суммы (последний из массива «c» будет первым в массиве rezc).

В цикле do 1122 идет пересылка в массив rezdr дробной части суммы (последний из дробной части будет первым в rezdr).

Далее идет анализ длин целой и дробной части результата суммирования.

2.2 Подпрограмма по реализации сложения с учетом знаков слагаемых

Вызов подпрограммы(приложение-1):

call_slzn(za, zb, zc, n, nd, a, ad, m, md, b, bd, kcel, kdro, rezc, rezdr, krazr),

za – знак первого слагаемого,

zb – знак второго слагаемого,

zc – знак суммы,

остальные параметры аналогичны параметрам при вызове подпрограммы Sl.

В подпрограмме slzn идет сначала запоминание исходных значений слагаемых.

В переменной ip – произведение знаков слагаемых.

Далее идет анализ знаков слагаемых, длин целых частей слагаемых.

По результатам анализа идет выбор операций.

В цикле do 786 идет анализ целых частей слагаемых при равенстве длин целых частей.

Далее идет нахождение минимальной длины дробной части двух слагаемых.

В цикле do 791 идет анализ дробных частей слагаемых при равенстве целых частей слагаемых.

В операторе if(nd.le.md) goto 795 идет анализ длин дробных частей слагаемых.

Если первое слагаемое положительное, то после метки 782 идут пересылки:

a↔b,

ad↔bd,

n↔m,

nd↔md,

za↔zb

После метки 785 идет восстановление исходных значений слагаемых.

2.3 Подпрограмма по реализации вычитания без учета знаков уменьшаемого и вычитаемого

Вызов подпрограммы(приложение-1):

call wich (n,nd,a,ad,m,md,b,bd,kcra,kdra,cra,dra,krazr),

n- количество разрядов в целой части уменьшаемого,

nd- количество разрядов в дробной части уменьшаемого,

a - массив разрядов целой части уменьшаемого,

ad - массив разрядов дробной части уменьшаемого,

m - количество разрядов в целой части вычитаемого,

md – количество разрядов в дробной части вычитаемого,

b – массив разрядов целой части вычитаемого,

bd – массив разрядов дробной части вычитаемого,

kcra - количество разрядов в целой части разности,

kdra – количество разрядов в дробной части разности,

cra – массив разрядов целой части разности,

dra – массив разрядов дробной части разности,

krazr - количество разрядов в дробной части длинной арифметики.

В цикле do 56881 идет обнуление массивов целой и дробной частей разности, результата вычитания.

В операторе if(n.ge.m) goto 897 идет сравнение длин целых частей уменьшаемого и вычитаемого.

Если количество разрядов в целой части уменьшаемого меньше чем количество разрядов в целой части вычитаемого, то осуществляем пересылки целых и дробных частей уменьшаемого и вычитаемого:

a↔b,

ad↔bd,

n↔m,

nd↔md

Реализация пересылок с оператора do 678 по метку 897.

После метки 897 находим максимальное количество разрядов для дробной части результата вычитания, разности.

В цикле do 79 идет расширение дробной части вычитаемого.

В цикле do 82 идет расширение дробной части уменьшаемого.

В цикле do 100 идет формирование единого массива уменьшаемого (целая + дробная часть).

В цикле do 101 идет формирование единого массива вычитаемого (целая + дробная часть).

Далее в подпрограмме:

i – номер разряда результата вычитания, разности,

nt – номер разряда в уменьшаемом,

mt – номер разряда в вычитаемом.

В операторе if(a(nt).ge.b(mt)) goto 1 идет сравнение разрядов в уменьшаемом и вычитаемом.

С метки 3 по оператор goto 3 идет поиск ненулевого разряда, который старше разряда с номером nt в массиве «a» (это когда a(nt)<b(mt)).

После метки 2 в операторе

a(np)=a(np)-1

идет уменьшение ненулевого разряда в массиве «a» на 1 (занимание единицы при вычитании).

С метки 7 по оператор goto 7 идет заполнение нулевых разрядов в массиве «a» от np+1 разряда до nt-1 разряда девятками.

После метки 30 идет прибавление 10 к разряду a(nt) в уменьшаемом.

После метки 1 идет увеличение счетчика разрядов в результате вычитания,разности, и присваивание

c(i)=a(nt)-b(mt)

Далее уменьшение на 1 номеров разрядов nt, mt, анализ значений mt,n,m, заполнение массива результата вычитания разрядами массива «a», когда их больше по количеству чем в массиве «b» (количество разрядов в целых частях).

После метки 999:

kcra – количество разрядов в целой части разности,

kdra – количество разрядов в дробной части разности,

im – количество ведущих нулей в разности (0001).

С оператора do 2001 по метку 973 идет формирование массива cra целой части результата вычитания.

После метки 973

kcra=kcra-im-количество разрядов в целой части разности после исключения нулей в старших разрядах.

В цикле do 2002 идет формирование массива разрядов дробной части разности dra.

Далее идет уточнение количества разрядов в целой и дробной частях результата вычитания.

2.4 Подпрограмма по реализации вычитания с учетом знаков уменьшаемого и вычитаемого

Вызов подпрограммы(приложение-1):

call wiczn(za, zb, zc, n, nd, a, ad, m, md, b, bd, kcel, kdra, rezc, rezdr, krazr),

za – знак уменьшаемого,

zb – знак вычитаемого,

zc – знак разности,

остальные параметры аналогичны параметрам подпрограммы wich при вызове.

Сначала в подпрограмме wiczn происходит запоминание исходных значений уменьшаемого и вычитаемого.

Потом в операторе

zb=zb×(-1)

происходит смена знака вычитаемого.

После этого далее по алгоритму подпрограмма wiczn полностью аналогична подпрограмме slzn.

2.5 Подпрограмма по реализации умножения без учета знаков

Вызов подпрограммы(приложение-1):

call ymn (n, nd, a, ad, m, md, b, bd, kcel, kdro, ctot, cdtot, krazr),

n – количество разрядов в целой части первого сомножителя,

nd - количество разрядов в дробной части первого сомножителя,

a – массив разрядов целой части первого сомножителя,

ad – массив разрядов дробной части первого сомножителя,

m – количество разрядов в целой части второго сомножителя,

md – количество разрядов в дробной части второго сомножителя,

b – массив разрядов целой части второго сомножителя,

bd – массив разрядов дробной части второго сомножителя,

kcel – количество разрядов в целой части произведения,

kdro – количество разрядов в дробной части произведения,

ctot – массив разрядов целой части произведения,

cdtot – массив разрядов дробной части произведения,

krazr – количество разрядов в дробной части длинной арифметики.

В цикле do 66987 идет обнуление целой и дробной частей сумматора.

В цикле do 100 идет формирование массива «a» для первого сомножителя (целая + дробная часть).

В цикле do 101 идет формирование массива “b” для второго сомножителя (целая + дробная часть).

Далее:

ntot – количество разрядов в целой части сумматора,

kdtot - количество разрядов в дробной части сумматора.

В цикле do 651 идет обнуление целой части сумматора.

В цикле do 652 идет обнуление дробной части сумматора.

Далее:

i – счетчик разрядов в произведении,

im – счетчик количества разрядов для определения конца дробной части произведения и начала целой части произведения,

p – значение переноса из младшего разряда в старший разряд,

nt – номер разряда в массиве «a» первого сомножителя,

mt – номер разряда в массиве «b» второго сомножителя.

В цикле do 941 идет обнуление результата произведения всех разрядов первого сомножителя на один разряд второго сомножителя.

С метки 701 по метку 1701 идет поразрядное умножение первого сомножителя «a» (все разряды) на один разряд b(mt) второго сомножителя.

Результат произведения в массиве c(i).

После метки 1701 и до метки 8722 идет формирование целой (массив csl) и дробной (массив csld) частей результата умножения первого множителя (всех разрядов) на разряд второго множителя.

После метки 8722 подпрограмма sl осуществляет суммирование результатов умножения (массивы csl, csld) первого множителя (всех разрядов) на разряд второго множителя.

В качестве второго слагаемого в sl идет сумматор (массивы ctot, cdtot).

Результат суммирования в массивах rezc, rezdr.

После суммирования идет переопределение сумматора в циклах do 704 и do 705.

Далее изменяются параметры сумматора ntot, kdtot.

После этого идет уменьшение номера разряда во втором сомножителе и так в цикле для всех разрядов второго сомножителя осуществляется их умножение на первый сомножитель (все разряды) и суммация этих произведений.

Результат произведения сомножителей будет в массивах ctot, cdtot с параметрами kcel (длина целой части произведения), kdro (длина дробной части произведения).

Далее в циклах do 7766 и do 7768 идет формирование целой части результата умножения.

Убираются ведущие нули 0001.

Далее идет уточнение количества разрядов в целой и дробной частях произведения.

2.6 Подпрограмма умножения с учетом знаков сомножителей

Вызов подпрограммы(приложение-1):

call ymnzn (za, zb, zc, n, nd, a, ad, m, md, b, bd, kcel, kdro, ctot, cdtot, krazr),

za – знак первого сомножителя,

zb – знак второго сомножителя,

zc – знак произведения.

Остальные параметры полностью аналогичны параметрам при вызове подпрограммы ymn.

Сначала в подпрограмме идет запоминание исходных значений сомножителей.

Потом определение знака произведения:

zc=za×zb

Далее идет вызов подпрограммы ymn по реализации длинного умножения.

Потом идет восстановление исходных значений сомножителей.

2.7 Подпрограмма по реализации деления без учета знаков

Вызов подпрограммы(приложение-1):

call deld (n, nd, a, ad, m, md, b, bd, kcdel, kddel, cdel, ddel, krazr),

n – количество разрядов в целой части делимого,

nd – количество разрядов в дробной части делимого,

a – массив разрядов целой части делимого,

ad – массив разрядов дробной части делимого,

m – количество разрядов в целой части делителя,

md – количество разрядов в дробной части делителя,

b – массив разрядов целой части делителя,

bd – массив разрядов дробной части делителя,

kcdel – количество разрядов в целой части частного,

kddel – количество разрядов в дробной части частного,

cdel – массив разрядов целой части частного,

ddel – массив разрядов дробной части частного,

krazr – количество разрядов в дробной части длинной арифметики.

В цикле do 66789 идет обнуление результатов деления, массивов cdel, ddel.

Далее находится максимальная длина для дробной части.

В цикле do 79 идет добавление нулей в дробной части делителя.

В цикле do 82 идет добавление нулей в дробной части делимого.

В цикле do 100 формируем массив “a” делимого (целая + дробная часть).

В циклах do 5671 и do 5673 убираем ведущие нули в массиве “a” (~~000~~1).

В циклах do 101, do 15671, do 15673 аналогично идет формирование массива “b”, делителя.

В цикле do 2677 идет обнуление сумматора, массивы ctot, cdtot.

Далее:

kctot – количество разрядов в целой части сумматора,

kdto – количество разрядов в дробной части сумматора,

ce – целая часть единичного значения,

kedi - количество разрядов в целой части единичного значения,

ced – дробная часть единичного значения,

kedid – количество разрядов в дробной части единичного значения,

kad – количество разрядов в дробной части делимого после преобразования его в один массив “a” (целая + дробная часть исходного значения),

kbd – количество разрядов в дробной части делителя после преобразования его в один массив “b” (целая + дробная часть исходного значения),

ad – дробная часть делимого после его преобразования в один массив “a” (целая + дробная часть исходного значения),

bd – дробная часть делителя после его преобразования в один массив “b” (целая + дробная часть исходного значения),

nat – количество разрядов в делимом,

mat – количество разрядов в делителе,

isd – счетчик разрядов в дробной части частного,

kcel – количество разрядов в целой части результата суммирования (подпрограмма sl),

kdro – количество разрядов в дробной части результата суммирования (подпрограмма sl),

rezc – массив разрядов целой части результата суммирования (подпрограмма sl),

rezdr – массив разрядов дробной части результата суммирования (подпрограмма sl).

С метки 2010 по метку 1231 реализуется алгоритм вычитания из делимого делителя и подсчета количества этих вычитаний, то есть расчет значения частного.

После метки 1231 идет корректировка количества разрядов в целой и дробной частях частного.

2.8 Подпрограмма по реализации деления с учетом знаков.

Вызов подпрограммы(приложение-1):

call delzn (za, zb, zc, n, nd, a, ad, m, md, b, bd, kcdel, kddel, cdel, ddel, krazr),

za – знак делимого,

zb – знак делителя.

zc – знак частного,

остальные параметры аналогичны параметрам подпрограммы deld при вызове.

Вначале подпрограммы идет запоминание исходных данных,значений делимого и делителя.

Потом идет вычисление знака частного:

zc=za×zb

Потом идет вызов подпрограммы deld для реализации длинного деления.

Потом идет восстановление исходных данных, делимого и делителя.

2.9 Главная программа chispi.for,pikontr.for

В программе **chispi.for** осуществляется(приложение-1) ввод исходных данных,реализация метода прямоугольников для вычисления определенного интеграла в длинной арифметике без имитации распараллеливания процедуры суммирования,запись результатов вычислений в файл.

В программе **pikontr.for** осуществляется(приложение-1) ввод исходных данных из файла kontrpi,реализация метода прямоугольников для вычисления определенного интеграла в длинной арифметике без имитации распараллеливания процедуры суммирования с созданием контрольных точек,запись результатов вычислений в файл.

2.10 Главная программа pidlpar1.for

Осуществляется(приложение-1) ввод исходных данных,реализация метода прямоугольников для вычисления определенного интеграла в длинной арифметике с имитацией распараллеливания процедуры суммирования,запись результатов вычислений в файл.

СПИСОК ЛИТЕРАТУРЫ

1. Й.Ленгсам, М.Огенстайн, А.Тененбаум
Структуры данных для персональных ЭВМ//Издательство "Мир", Москва,1989.

2. A.Aho, J.Hopcroft, J.Ullman
Data Structures and Algorithms//Addison-Wesley, Reading, Mass, 1982.

3. Ван Тассел Д.
Стиль, разработка, эффективность, отладка и испытание программ // Издательство "Мир", Москва, 1981.

4. Дал У., Дейкстра Э., Хоор К.
Структурное программирование// М.: Мир, 1975.

5. Ахо А., Хопкрофт Дж., Ульман Дж.
Построение и анализ вычислительных алгоритмов//М.:Мир, 1979.

6. Р.Джордейн
Справочник программиста персональных компьютеров типа IBM PC, XT и AT//Москва,"Финансы и статистика", 1992.

7. Фирма "Диалектика" TURBO VISION для C++ // Киев, 1992.

8. Мак-Кракен, Дорн
Численные методы и программирование на Фортране // Издательство "Мир", Москва, 1982.

9. Д.Стэбли
Логическое программирование в системе IBM/360. // Издательство "Мир", Москва, 1979.

10. У.Радд
Программирование на языке ассемблера и вычислительные системы IBM 360 и 370 // Издательство "Мир", Москва,1979.

11.Дж.Донован
Системное программирование // Москва,"Мир",1976.

ПРИЛОЖЕНИЕ-1(наборы с исходными текстами программ на Фортране)
Набор chispi.for(вычисление числа пи без имитации распараллеливания процедуры суммирования)

```
$large
      dimension ixc(400),ixd(400),isuc(400),isud(400),idxc(400)
      dimension idxd(400),icrez(400),idrez(400),icr1(400)
      dimension idr1(400),icr2(400),idr2(400),iced(400),ided(400)
      dimension ic4(400),id4(400)
c-kcx-количество разрядов в целой части x
c-kdx-количество разрядов в дробной части x
c-kcsu-количество разрядов в целой части интегральной суммы
c-kdsu-количество разрядов в дробной части интегральной суммы
c-ixc-массив разрядов целой части x
c-ixd-массив разрядов дробной части x
c-isuc-массив разрядов целой части интегральной суммы
c-isud-массив разрядов дробной части интегральной суммы
c-idxc-массив разрядов целой части dx
c-idxd-массив разрядов дробной части dx
c-kcd-количество разрядов в целой части dx
c-kdd-количество разрядов в дробной части dx
c-iced-массив разрядов целой части 1(единицы)
c-ided-массив разрядов дробной части 1(единицы)
c-kced-количество разрядов в целой части 1(единицы)
c-kded-количество разрядов в дробной части 1(единицы)
      write(*,*) 'введите количество разрядов в шаге dx'
      read(*,*) kdd
      write(*,*) 'введите количество разрядов в дробной части'
      read(*,*) krazr
c     krazr=10
c-определяем 1(единицу)
      izn1=1
      kced=1
      kded=1
      iced(1)=1
      ided(1)=0
c- определяем dx
      izndx=1
      kcd=1
c     kdd=2
      idxc(1)=0
      do 1 i=1,kdd
1     idxd(i)=0
      idxd(kdd)=1
c-определяем 4(четыре)
      izn4=1
      kc4=1
      kd4=1
      ic4(1)=4
      id4(1)=0
c-определяем начальное x(x=0)
      iznx=1
      kcx=1
      kdx=1
      ixc(1)=0
      ixd(1)=0
c-определяем начальное значение интегральной суммы
      iznsu=1
      kcsu=1
      kdsu=1
      isuc(1)=0
      isud(1)=0
```

```
      isch=0

22    continue
      isch=isch+1
      if(isch.lt.1000) goto 933
      write(*,*) 'значение x(дробная часть)'
      write(*,34) (ixd(j),j=1,kdx)
      isch=0
933   continue
c-умножаем x на x
      call ymnzn(iznx,iznx,izrez,
     *kcx,kdx,ixc,ixd,
     *kcx,kdx,ixc,ixd,
     *kcrez,kdrez,icrez,idrez,krazr)
c-складываем 1+x*x
      call slzn(izn1,izrez,iz1,
     *kced,kded,iced,ided,
     *kcrez,kdrez,icrez,idrez,
     *ic1,id1,icr1,idr1,krazr)
c-делим 1/(1+x*x)
      call delzn(izn1,iz1,izdel,
     *kced,kded,iced,ided,
     *ic1,id1,icr1,idr1,
     *ic2,id2,icr2,idr2,krazr)
c-умножаем 1/(1+x*x) на dx
      call ymnzn(izdel,izndx,izymn,
     *ic2,id2,icr2,idr2,
     *kcd,kdd,idxc,idxd,
     *ic1,id1,icr1,idr1,krazr)
c-наращиваем интегральную сумму
      call slzn(iznsu,izymn,izsum,
     *kcsu,kdsu,isuc,isud,
     *ic1,id1,icr1,idr1,
     *ic2,id2,icr2,idr2,krazr)
c-переопределяем интегральную сумму
      iznsu=izsum
      kcsu=ic2
      kdsu=id2
      do 2 i=1,ic2
2     isuc(i)=icr2(i)
      do 3 i=1,id2
3     isud(i)=idr2(i)
c- увеличиваем x на dx
      call slzn(iznx,izndx,izrez,
     *kcx,kdx,ixc,ixd,
     *kcd,kdd,idxc,idxd,
     *kcrez,kdrez,icrez,idrez,krazr)
c- переопределяем x
      kcx=kcrez
      kdx=kdrez
      do 5 i=1,kcrez
5     ixc(i)=icrez(i)
      do 6 i=1,kdrez

6     ixd(i)=idrez(i)
c-сравнение с единицей
      if(ixc(1).ge.iced(1)) goto 11
      goto 22
11    continue
c-уьножение интегральной суммы на 4
      call ymnzn(iznsu,izn4,izrez,
     *kcsu,kdsu,isuc,isud,
     *kc4,kd4,ic4,id4,
```

```
     *kcrez,kdrez,icrez,idrez,krazr)

c-вывод числа пи
      open(6,file='ch_pi',status='Unknown')
      write(6,41)
41    format(/1x,'шаг интегрирования')
      write(6,42)
42    format(/1x,'целая часть шага')
      write(6,34) (idxc(i),i=1,kcd)
      write(6,43)
43    format(/1x,'дробная часть шага')
      write(6,34) (idxd(i),i=1,kdd)

      write(6,31)
31    format(/1x,'число пи')
      write(6,33)
      write(*,33)
33    format(/1x,'целая часть')
      write(6,34) (icrez(i),i=1,kcrez)
      write(*,34) (icrez(i),i=1,kcrez)
34    format(/1x,70i1)
      write(6,35)
      write(*,35)
35    format(/1x,'дробная часть')
      write(6,34) (idrez(i),i=1,kdrez)
      write(*,34) (idrez(i),i=1,kdrez)
      pause
      close(6,status='keep')
      end
```

Набор pikontr.for(вычисление числа пи с созданием контрольных точек)

```
$large
      dimension ixc(400),ixd(400),isuc(400),isud(400),idxc(400)
      dimension idxd(400),icrez(400),idrez(400),icr1(400)
      dimension idr1(400),icr2(400),idr2(400),iced(400),ided(400)
      dimension ic4(400),id4(400)
      open(5,file='kontrpi',status='old')
c-kcx-количество разрядов в целой части x
c-kdx-количество разрядов в дробной части x
c-kcsu-количество разрядов в целой части интегральной суммы
c-kdsu-количество разрядов в дробной части интегральной суммы
c-ixc-массив разрядов целой части x
c-ixd-массив разрядов дробной части x
c-isuc-массив разрядов целой части интегральной суммы
c-isud-массив разрядов дробной части интегральной суммы
c-idxc-массив разрядов целой части dx
c-idxd-массив разрядов дробной части dx
c-kcd-количество разрядов в целой части dx
c-kdd-количество разрядов в дробной части dx
c-iced-массив разрядов целой части 1(единицы)
c-ided-массив разрядов дробной части 1(единицы)
c-kced-количество разрядов в целой части 1(единицы)
c-kded-количество разрядов в дробной части 1(единицы)
c     write(*,*) 'введите количество разрядов в шаге dx'
      read(5,*) kdd,krazr,kshag
c     write(*,*) 'введите количество разрядов в дробной части'
c     read(*,*) krazr
c     krazr=10
c-определяем 1(единицу)
      izn1=1
      kced=1
      kded=1
      iced(1)=1
      ided(1)=0
c- определяем dx
      izndx=1
      kcd=1
c     kdd=2
      idxc(1)=0
      do 1 i=1,kdd
1     idxd(i)=0
      idxd(kdd)=1
c-определяем 4(четыре)
      izn4=1
      kc4=1
      kd4=1
      ic4(1)=4
      id4(1)=0
c-определяем начальное x(x=0)
      iznx=1
      kcx=1
      kdx=1
      ixc(1)=0
      ixd(1)=0
      read(5,*) iznx,kcx,kdx
      read(5,9201)(ixc(i),i=1,kcx)
9201  format(60i1)
      read(5,9201) (ixd(i),i=1,kdx)
c-определяем начальное значение интегральной суммы
      iznsu=1
      kcsu=1
      kdsu=1
```

```
      isuc(1)=0
      isud(1)=0
      read(5,*) iznsu,kcsu,kdsu
      read(5,9201) (isuc(i),i=1,kcsu)
      read(5,9201) (isud(i),i=1,kdsu)
      close(5,status='keep')
      isch=0
22    continue
      isch=isch+1
      if(isch.lt.10000) goto 933
      write(*,*) 'создается контрольная точка,снимать нельзя!'
      open(5,file='kontrpi',status='Unknown')
      write(5,*) kdd,krazr,kshag
      write(5,*) iznx,kcx,kdx
      write(5,9201)(ixc(i),i=1,kcx)
      write(5,9201) (ixd(i),i=1,kdx)
      write(5,*) iznsu,kcsu,kdsu
      write(5,9201) (isuc(i),i=1,kcsu)
      write(5,9201) (isud(i),i=1,kdsu)
      close(5,status='keep')

      write(*,*) 'снимать можно!'
c     read(*,*) isnjt

c     if(isnjt.eq.0) stop

      isch=0
933   continue
c-умножаем x на x
      call ymnzn(iznx,iznx,izrez,
     *kcx,kdx,ixc,ixd,
     *kcx,kdx,ixc,ixd,
     *kcrez,kdrez,icrez,idrez,krazr)
c-складываем 1+x*x
      call slzn(izn1,izrez,iz1,
     *kced,kded,iced,ided,
     *kcrez,kdrez,icrez,idrez,
     *ic1,id1,icr1,idr1,krazr)
c-делим 1/(1+x*x)
      call delzn(izn1,iz1,izdel,
     *kced,kded,iced,ided,
     *ic1,id1,icr1,idr1,
     *ic2,id2,icr2,idr2,krazr)
c-умножаем 1/(1+x*x) на dx
      call ymnzn(izdel,izndx,izymn,
     *ic2,id2,icr2,idr2,
     *kcd,kdd,idxc,idxd,
     *ic1,id1,icr1,idr1,krazr)
c-наращиваем интегральную сумму
      call slzn(iznsu,izymn,izsum,
     *kcsu,kdsu,isuc,isud,
     *ic1,id1,icr1,idr1,
     *ic2,id2,icr2,idr2,krazr)
c-переопределяем интегральную сумму
      iznsu=izsum
      kcsu=ic2
      kdsu=id2
      do 2 i=1,ic2
2     isuc(i)=icr2(i)
      do 3 i=1,id2
3     isud(i)=idr2(i)
c- увеличиваем x на dx
      call slzn(iznx,izndx,izrez,
```

```
     *kcx,kdx,ixc,ixd,
     *kcd,kdd,idxc,idxd,
     *kcrez,kdrez,icrez,idrez,krazr)
c- переопределяем x
      kcx=kcrez
      kdx=kdrez
      do 5 i=1,kcrez
5     ixc(i)=icrez(i)
      do 6 i=1,kdrez
6     ixd(i)=idrez(i)
c-сравнение с единицей
      if(ixc(1).ge.iced(1)) goto 11
      goto 22
11    continue
c-уьножение интегральной суммы на 4
      call ymnzn(iznsu,izn4,izrez,
     *kcsu,kdsu,isuc,isud,
     *kc4,kd4,ic4,id4,
     *kcrez,kdrez,icrez,idrez,krazr)
c-вывод числа пи
      open(6,file='ch_pi',status='Unknown')
      write(6,41)
41    format(/1x,'шаг интегрирования')
      write(6,42)
42    format(/1x,'целая часть шага')
      write(6,34) (idxc(i),i=1,kcd)
      write(6,43)
43    format(/1x,'дробная часть шага')
      write(6,34) (idxd(i),i=1,kdd)

      write(6,31)
31    format(/1x,'число пи')
      write(6,33)
      write(*,33)
33    format(/1x,'целая часть')
      write(6,34) (icrez(i),i=1,kcrez)
      write(*,34) (icrez(i),i=1,kcrez)
34    format(/1x,70i1)
      write(6,35)
      write(*,35)
35    format(/1x,'дробная часть')
      write(6,34) (idrez(i),i=1,kdrez)
      write(*,34) (idrez(i),i=1,kdrez)
      pause
      close(6,status='keep')
      end
```

Набор pidlpar1.for(вычисление числа пи с имитацией распараллеливания процедуры суммирования)

```
$large
      dimension ixc(400),ixd(400),isuc(400),isud(400),idxc(400)
      dimension idxd(400),icrez(400),idrez(400),icr1(400)
      dimension idr1(400),icr2(400),idr2(400),iced(400),ided(400)
      dimension ic4(400),id4(400),icpr(400),idpr(400),ickop(400)
      dimension idkop(400),idxn(400),icxn(400),irab(400)
      open(6,file='pipardl',status='Unknown')
c-kcx-количество разрядов в целой части x
c-kdx-количество разрядов в дробной части x
c-kcsu-количество разрядов в целой части интегральной суммы
c-kdsu-количество разрядов в дробной части интегральной суммы
c-ixc-массив разрядов целой части x
c-ixd-массив разрядов дробной части x
c-isuc-массив разрядов целой части интегральной суммы
c-isud-массив разрядов дробной части интегральной суммы
c-idxc-массив разрядов целой части dx
c-idxd-массив разрядов дробной части dx
c-kcd-количество разрядов в целой части dx
c-kdd-количество разрядов в дробной части dx
c-iced-массив разрядов целой части 1(единицы)
c-ided-массив разрядов дробной части 1(единицы)
c-kced-количество разрядов в целой части 1(единицы)
c-kded-количество разрядов в дробной части 1(единицы)
      write(*,*) 'введите количество разрядов в шаге dx'
      read(*,*) kdd
      write(*,*) 'введите количество разрядов в дробной части'
      read(*,*) krazr
      write(*,*) 'введите количество процессов'
      read(*,*) kproc
      write(6,873) kdd,krazr,kproc
873   format(/1x,'количество разрядов в шаге dx=',i10/1x,
     *'количество разрядов в дробной части=',i5/1x,
     *'количество процессов=',i2/)
c-количество процессов
      izkop=1
      kdkop=1
c-формирование массива целой части количества процессов
      ikp=0
2001  continue
      ios=kproc-kproc/10*10
      ikp=ikp+1
      irab(ikp)=ios
      kproc=kproc/10
      if(kproc.ne.0) goto 2001
      kckop=ikp
      l=0
      do 2002 i=ikp,1,-1
      l=l+1
      ickop(l)=irab(i)
c     write(6,6781) ickop(l)
6781  format(/1x,'ickop=',i1/)
2002  continue
c     ickop(1)=kproc
      idkop(1)=0
c-определяем 1(единицу)
      izn1=1
      kced=1
      kded=1
      iced(1)=1
      ided(1)=0
```

```
c- определяем dx
      izndx=1
      kcd=1
c     kdd=2
      idxc(1)=0
      do 1 i=1,kdd
1     idxd(i)=0
      idxd(kdd)=1
c-определяем 4(четыре)
      izn4=1
      kc4=1
      kd4=1
      ic4(1)=4
      id4(1)=0
c-определяем начальное x(x=0)
      iznx=1
      kcx=1
      kdx=1
      ixc(1)=0
      ixd(1)=0
c-меняющееся начальное x
      iznax=1
      kcxn=1
      kdxn=1
      icxn(1)=0
      idxn(1)=0
c-определяем начальное значение интегральной суммы
      iznsu=1
      kcsu=1
      kdsu=1
      isuc(1)=0
      isud(1)=0
      isch=0
c-определяем начальный номер процесса
      iznpr=1
      kcpr=1
      kdpr=1
      icpr(1)=1
      idpr(1)=0
777   continue
c-вычитаем из номера процесса единицу

      call wiczn(iznpr,izn1,izrez,
     *kcpr,kdpr,icpr,idpr,
     *kced,kded,iced,ided,
     *ic1,id1,icr1,idr1,krazr)
c-умножаем (npr-1)*dx
      call ymnzn(izrez,izndx,iznak,
     *ic1,id1,icr1,idr1,
     *kcd,kdd,idxc,idxd,
     *ic2,id2,icr2,idr2,krazr)
c-складываем x+(npr-1)*dx
      call slzn(iznax,iznak,izrez,
     *kcxn,kdxn,icxn,idxn,
     *ic2,id2,icr2,idr2,
     *ic1,id1,icr1,idr1,krazr)
c-переопределение x
      iznx=izrez
      kcx=ic1
      kdx=id1
      do 210 i=1,kcx
210   ixc(i)=icr1(i)
      do 211 i=1,kdx
```

```
211   ixd(i)=idr1(i)
      isch=isch+1
      if(isch.lt.1000) goto 678
      if(ixc(1).ge.iced(1)) goto 1111
      write(*,891)
891   format(/1x,'номер процесса')
      write(*,894) (icpr(l),l=1,kcpr)
894   format(1x,50i1)
c     write(*,*) 'дробная часть x'
c     write(*,34) (ixd(l),l=1,kdx)
      isch=0
678   continue
c-сравниваем x с единицей
      if(ixc(1).ge.iced(1)) goto 1111
c-умножаем x на x
      call ymnzn(iznx,iznx,izrez,
     *kcx,kdx,ixc,ixd,
     *kcx,kdx,ixc,ixd,
     *kcrez,kdrez,icrez,idrez,krazr)
c-складываем 1+x*x
      call slzn(izn1,izrez,iz1,
     *kced,kded,iced,ided,
     *kcrez,kdrez,icrez,idrez,
     *ic1,id1,icr1,idr1,krazr)
c-делим 1/(1+x*x)
      call delzn(izn1,iz1,izdel,
     *kced,kded,iced,ided,
     *ic1,id1,icr1,idr1,
     *ic2,id2,icr2,idr2,krazr)
c-умножаем 1/(1+x*x) на dx
      call ymnzn(izdel,izndx,izymn,
     *ic2,id2,icr2,idr2,
     *kcd,kdd,idxc,idxd,
     *ic1,id1,icr1,idr1,krazr)
c-наращиваем интегральную сумму
      call slzn(iznsu,izymn,izsum,
     *kcsu,kdsu,isuc,isud,
     *ic1,id1,icr1,idr1,
     *ic2,id2,icr2,idr2,krazr)
c-переопределяем интегральную сумму
      iznsu=izsum
      kcsu=ic2
      kdsu=id2
      do 2 i=1,ic2
2     isuc(i)=icr2(i)
      do 3 i=1,id2
3     isud(i)=idr2(i)
c-увеличить меняющееся начальное x на kproc*dx
c-вычислить kproc*dx
      call ymnzn(izkop,izndx,izrez,
     *kckop,kdkop,ickop,idkop,
     *kcd,kdd,idxc,idxd,
     *ic1,id1,icr1,idr1,krazr)
c-суммируем меняющееся начальное x+kproc*dx
      call slzn(iznax,izrez,iz1,
     *kcxn,kdxn,icxn,idxn,
     *ic1,id1,icr1,idr1,
     *ic2,id2,icr2,idr2,krazr)
c-переопределяем меняющееся начальное x
      iznax=iz1
      kcxn=ic2
      kdxn=id2
      do 567 i=1,kcxn
```

```
567   icxn(i)=icr2(i)
      do 568 i=1,kdxn
568   idxn(i)=idr2(i)
      goto 777
1111  continue
c-увеличиваем номер процесса на один
      call slzn(iznpr,izn1,izrez,
     *kcpr,kdpr,icpr,idpr,
     *kced,kded,iced,ided,
     *ic1,id1,icr1,idr1,krazr)
c-переопределяем номер процесса
      iznpr=izrez
      kcpr=ic1
      kdpr=id1
      do 250 i=1,kcpr
250   icpr(i)=icr1(i)
      do 251 i=1,kdpr
251   idpr(i)=idr1(i)
c-сравнение номера процесса с количеством процессов
      if(kcpr.gt.kckop) goto 11
      if(kcpr.lt.kckop) goto 4002
      do 4001 i=1,kcpr
      if(icpr(i).gt.ickop(i)) goto 11
4001  continue
4002  continue
      iznax=1
      kcxn=1
      kdxn=1
      icxn(1)=0
      idxn(1)=0
      goto 777
11    continue
c-умножение интегральной суммы на 4
      call ymnzn(iznsu,izn4,izrez,
     *kcsu,kdsu,isuc,isud,
     *kc4,kd4,ic4,id4,
     *kcrez,kdrez,icrez,idrez,krazr)
c-вывод числа пи
c     open(6,file='ch_pi',status='Unknown')
      write(6,41)
41    format(/1x,'шаг интегрирования')
      write(6,42)
42    format(/1x,'целая часть шага')
      write(6,34) (idxc(i),i=1,kcd)
      write(6,43)
43    format(/1x,'дробная часть шага')
      write(6,34) (idxd(i),i=1,kdd)
      write(6,31)
31    format(/1x,'число пи')
      write(6,33)
      write(*,33)
33    format(/1x,'целая часть')
      write(6,34) (icrez(i),i=1,kcrez)
      write(*,34) (icrez(i),i=1,kcrez)
34    format(1x,70i1)
      write(6,35)
      write(*,35)
35    format(/1x,'дробная часть')
      write(6,34) (idrez(i),i=1,kdrez)
      write(*,34) (idrez(i),i=1,kdrez)
      pause
      close(6,status='keep')
      end
```

Набор sl.for

```
c-длинное сложение
$large
      subroutine sl(n,nd,a,ad,m,md,b,bd,kcel,kdro,rezc,rezdr,krazr)
      dimension a(400),b(400),c(400),ad(400),bd(400),cd(400)
      dimension rezc(400),rezdr(400)
      integer a,b,c,rez,os,p,ad,bd,cd,rezc,rezdr
c     open(5,file='summod',status='Unknown')
c     write(*,111)
111   format(1x,'введите длину целой части первого слагаемого')
c     read(*,*) n
c     write(*,112)
112   format(1x,'введите длину дробной части первого слагаемого')
c     read(*,*) nd
c     write(*,1111)
1111   format(1x,'введите длину целой части второго слагаемого')
c     read(*,*) m
c     write(*,1112)
1112   format(1x,'введите длину дробной части второго слагаемого')
c     read(*,*) md


c      write(*,334)
334   format(1x,'введите через пробел разряды целой части',
     *' первого слагаемого')
c     read(*,*) (a(i),i=1,n)
c     write(*,335)
335   format(1x,'введите через пробел разряды дробной части',
     *' первого слагаемого')

c     read(*,*) (ad(i),i=1,nd)
c     write(*,1334)
1334   format(1x,'введите через пробел разряды целой части',
     *' второго слагаемого')

c     read(*,*) (b(i),i=1,m)
c     write(*,1335)
1335   format(1x,'введите через пробел разряды дробной части',
     *' второго слагаемого')


c     read(*,*) (bd(i),i=1,md)
c     write(5,*) 'целая часть первого слагаемого'
c     write(5,445) (a(i),i=1,n)
445   format(1x,70i1)
c     write(5,*) 'дробная часть первого слагаемого'
c     write(5,445) (ad(i),i=1,nd)
c     write(5,*) 'целая часть второго слагаемого'
c     write(5,445) (b(i),i=1,m)
c     write(5,*) 'дробная часть второго слагаемого'
c     write(5,445) (bd(i),i=1,md)
c     if(krazr.le.100) goto 66791
c     write(*,*) 'сложение,количество разрядов>100!'
c     pause
c     stop
66791 continue
        do 66875 i=1,400
      rezc(i)=0
66875 rezdr(i)=0
      if(n.ge.m) goto 897
      do 678 i=1,n
      c(i)=a(i)
```

```
678   continue
      do 679 i=1,nd
      cd(i)=ad(i)
679   continue
      do 680 i=1,m
      a(i)=b(i)
680   continue
      do 681 i=1,md
      ad(i)=bd(i)
681   continue
      do 682 i=1,n
      b(i)=c(i)
682   continue
      do 683 i=1,nd
      bd(i)=cd(i)
683   continue
      ir=n
      n=m
      m=ir
      ir=nd
      nd=md
      md=ir
897   continue
c-формирование двух массивов для сложения
      max=nd
      if(max.lt.md) max=md
      if(md.eq.max) goto 78
      in=md+1
      do 79 i=in,max
      bd(i)=0
79    continue
      goto 81
78    continue
      in=nd+1
      do 82 i=in,max
      ad(i)=0
82    continue
81    continue
      it=0
      in=n+1
      ik=n+max
      do 100 i=in,ik
      it=it+1
      a(i)=ad(it)
100   continue
      it=0
      in=m+1
      ik=m+max
      do 101 i=in,ik
      it=it+1
      b(i)=bd(it)
101   continue
c-сложение
      i=0
      p=0
      nt=n+max
      mt=m+max
7     continue
      if(mt.lt.1) rez=a(nt)
      if(mt.ge.1) rez=a(nt)+b(mt)
      rez=rez+p
      i1=rez/10
      os=rez-i1*10
```

```
      i=i+1
      c(i)=os
c     if(i.ne.1) goto 1
      p=i1
      nt=nt-1
      mt=mt-1
      if((p.eq.0).and.(mt.lt.1)) goto 4
      if((nt.lt.1).and.(p.ne.0)) goto 5
      if((nt.lt.1).and.(p.eq.0)) goto 4
      goto 7
5     continue
      i=i+1
      c(i)=p
      goto 67
4     continue
89    continue
      if(nt.lt.1) goto 67
      i=i+1
      c(i)=a(nt)
      nt=nt-1
      goto 89
67    continue
c     write(5,*) 'целая часть суммы'
      ik=max+1
c     write(5,445) (c(j),j=i,ik,-1)
c     write(5,*) 'дробная часть суммы'
c     write(5,445) (c(j),j=max,1,-1)
      kcel=i-max
      kdro=max
      it=0
      do 1121 j=i,ik,-1
      it=it+1
      rezc(it)=c(j)
1121  continue
      it=0
      do 1122 j=max,1,-1
      it=it+1
      rezdr(it)=c(j)
1122  continue
      if(kcel.ne.0) goto 971
      kcel=1
      rezc(1)=0
971   continue
      if(kdro.ne.0) goto 972
      kdro=1
      rezdr(1)=0
972   continue
      if(kdro.gt.krazr) kdro=krazr
c     if(kcel.le.krazr) goto 67291
c     write(*,67292) kcel,krazr
67292 format(1x,'сложение,количество разрядов в целой части=',i3,
     *' больше  заданного=',i3)
c     pause
c     stop
67291 continue
      return
      end
```

Набор slzn.for

```
$large
      subroutine slzn(za,zb,zc,n,nd,a,ad,m,md,b,bd,
     *kcel,kdro,rezc,rezdr,krazr)
c-длинное сложение с учетом знаков слагаемых
      dimension a(400),b(400),c(400),ad(400),bd(400),cd(400)
      dimension rezc(400),rezdr(400),ar(400),adr(400)
      dimension ai(400),adi(400),bi(400),bdi(400)
      integer a,b,c,rez,os,p,ad,bd,cd,rezc,rezdr,za,zb,zc
      integer ar,adr,ai,adi,bi,bdi,zais,zbis
c     open(5,file='dlsumzn',status='Unknown')
c     write(*,2001)
2001  format(1x,'введите знак первого слагаемого,1-плюс,-1-минус')
c     read(*,*) za
c     write(*,2002)
2002  format(1x,'введите знак второго слагаемого,1-плюс,-1-минус')
c     read(*,*) zb
c     write(*,111)
111   format(1x,'введите длину целой части первого слагаемого')
c     read(*,*) n
c     write(*,112)
112   format(1x,'введите длину дробной части первого слагаемого')
c     read(*,*) nd
c     write(*,1111)
1111   format(1x,'введите длину целой части второго слагаемого')
c     read(*,*) m
c     write(*,1112)
1112   format(1x,'введите длину дробной части второго слагаемого')
c     read(*,*) md



c      write(*,334)
334   format(1x,'введите через пробел разряды целой части',
     *' первого слагаемого')
c     read(*,*) (a(i),i=1,n)
c     write(*,335)
335   format(1x,'введите через пробел разряды дробной части',
     *' первого слагаемого')

c     read(*,*) (ad(i),i=1,nd)
c     write(*,1334)
1334   format(1x,'введите через пробел разряды целой части',
     *' второго слагаемого')

c     read(*,*) (b(i),i=1,m)
c     write(*,1335)
1335   format(1x,'введите через пробел разряды дробной части',
     *' второго слагаемого')


c     read(*,*) (bd(i),i=1,md)
c     write(5,*) 'целая часть первого слагаемого'
c     write(5,445) (a(i),i=1,n)
445   format(1x,70i1)
c     write(5,*) 'дробная часть первого слагаемого'
c     write(5,445) (ad(i),i=1,nd)
c     write(5,*) 'целая часть второго слагаемого'
c     write(5,445) (b(i),i=1,m)
c     write(5,*) 'дробная часть второго слагаемого'
c     write(5,445) (bd(i),i=1,md)
      zais=za
```

```
      zbis=zb
      do 5671 i=1,n
      ai(i)=a(i)
5671  continue
      do 5672 i=1,nd
      adi(i)=ad(i)
5672  continue
      do 5673 i=1,m
      bi(i)=b(i)
5673  continue
      do 5674 i=1,md
      bdi(i)=bd(i)
5674  continue
      ni=n
      ndi=nd
      mi=m
      mdi=md

      ip=za*zb
      if(ip.gt.0) goto 781
      if(za.eq.1) goto 782
1901  if(m.gt.n) goto 783
      if(m.eq.n) goto 784
      call wich(n,nd,a,ad,m,md,b,bd,kcel,kdro,rezc,rezdr,krazr)
      zc=za
      goto 785
784   continue
      do 786 i=1,m
      if(a(i).gt.b(i)) goto 787
      if(a(i).lt.b(i)) goto 788
786   continue
      nmin=nd
      if(nmin.gt.md) nmin=md
      do 791 i=1,nmin
      if(ad(i).gt.bd(i)) goto 792
      if(ad(i).lt.bd(i)) goto 793
791   continue
      if(nd.le.md) goto 795
      call wich(n,nd,a,ad,m,md,b,bd,kcel,kdro,rezc,rezdr,krazr)
      zc=za
      goto 785
795   continue
      call wich(m,md,b,bd,n,nd,a,ad,kcel,kdro,rezc,rezdr,krazr)
      zc=zb
      goto 785
793   continue
      call wich(m,md,b,bd,n,nd,a,ad,kcel,kdro,rezc,rezdr,krazr)
      zc=zb
      goto 785
792   continue
      call wich(n,nd,a,ad,m,md,b,bd,kcel,kdro,rezc,rezdr,krazr)
      zc=za
      goto 785
788   continue
      call wich(m,md,b,bd,n,nd,a,ad,kcel,kdro,rezc,rezdr,krazr)
      zc=zb
      goto 785
787   continue
      call wich(n,nd,a,ad,m,md,b,bd,kcel,kdro,rezc,rezdr,krazr)
      zc=za
      goto 785
783   continue
      call wich(m,md,b,bd,n,nd,a,ad,kcel,kdro,rezc,rezdr,krazr)
```

```
      zc=zb
      goto 785
782   continue
      do 1782 i=1,n
      ar(i)=a(i)
1782  continue
      do 1783 i=1,nd
      adr(i)=ad(i)
1783  continue
      nr=n
      ndr=nd
      do 1784 i=1,m
      a(i)=b(i)
1784  continue
      do 1785 i=1,md
      ad(i)=bd(i)
1785  continue
      n=m
      nd=md
      m=nr
      md=ndr
      do 1786 i=1,nr
      b(i)=ar(i)
1786  continue
      do 1787 i=1,ndr
      bd(i)=adr(i)
1787  continue
      zc=za
      za=zb
      zb=zc
      goto 1901
781   continue
      call sl(n,nd,a,ad,m,md,b,bd,kcel,kdro,rezc,rezdr,krazr)
      zc=za
785   continue
c     write(5,901) zc
901   format(1x,'знак суммы=',i2)
      n=ni
      nd=ndi
      m=mi
      md=mdi
      do 8721 i=1,n
      a(i)=ai(i)
8721  continue
      do 8722 i=1,nd
      ad(i)=adi(i)
8722  continue
      do 8723 i=1,m
      b(i)=bi(i)
8723  continue
      do 8724 i=1,md
      bd(i)=bdi(i)
8724  continue
      za=zais
      zb=zbis
      return
      end
```

Набор wich.for

```
c-длинное вычитание
$large
      subroutine wich(n,nd,a,ad,m,md,b,bd,kcra,kdra,cra,dra,krazr)
      dimension a(400),ad(400),b(400),bd(400),c(400),cd(400)
      dimension cra(400),dra(400)
      integer a,ad,b,bd,c,cd,cra,dra
c     open(5,file='dlinwich',status='Unknown')
c      write(*,111)
111   format(1x,'введите длину целой части уменьшаемого')
c     read(*,*) n
c     write(*,112)
112   format(1x,'введите длину дробной части уменьшаемого')
c     read(*,*) nd
c     write(*,1111)
1111   format(1x,'введите длину целой части вычитаемого')
c     read(*,*) m
c     write(*,1112)
1112   format(1x,'введите длину дробной части вычитаемого')
c     read(*,*) md


c      write(*,334)
334   format(1x,'введите через пробел разряды целой части',
     *' уменьшаемого')
c     read(*,*) (a(i),i=1,n)
c     write(*,335)
335   format(1x,'введите через пробел разряды дробной части',
     *' уменьшаемого')

c     read(*,*) (ad(i),i=1,nd)
c     write(*,1334)
1334   format(1x,'введите через пробел разряды целой части',
     *' вычитаемого')

c     read(*,*) (b(i),i=1,m)
c     write(*,1335)
1335   format(1x,'введите через пробел разряды дробной части',
     *' вычитаемого')


c     read(*,*) (bd(i),i=1,md)
c     write(5,*) 'целая часть уменьшаемого'
c     write(5,445) (a(i),i=1,n)
445   format(1x,79i1)
c     write(5,*) 'дробная часть уменьшаемого'
c     write(5,445) (ad(i),i=1,nd)
c     write(5,*) 'целая часть вычитаемого'
c     write(5,445) (b(i),i=1,m)
c     write(5,*) 'дробная часть вычитаемого'
c     write(5,445) (bd(i),i=1,md)
c     if(krazr.le.100) goto 66791
c     write(*,*) 'вычитание,количество разрядов>100!'
c     pause
c     stop
66791 continue
      do 56881 i=1,400
      cra(i)=0
56881 dra(i)=0
      if(n.ge.m) goto 897
      do 678 i=1,n
      c(i)=a(i)
```

```
678   continue
      do 679 i=1,nd
      cd(i)=ad(i)
679   continue
      do 680 i=1,m
      a(i)=b(i)
680   continue
      do 681 i=1,md
      ad(i)=bd(i)
681   continue
      do 682 i=1,n
      b(i)=c(i)
682   continue
      do 683 i=1,nd
      bd(i)=cd(i)
683   continue
      ir=n
      n=m
      m=ir
      ir=nd
      nd=md
      md=ir
897   continue

c-формирование двух массивов для вычитания
      max=nd
      if(max.lt.md) max=md
      if(md.eq.max) goto 78
      in=md+1
      do 79 i=in,max
      bd(i)=0
79    continue
      goto 81
78    continue
      in=nd+1
      do 82 i=in,max
      ad(i)=0
82    continue
81    continue
      it=0
      in=n+1
      ik=n+max
      do 100 i=in,ik
      it=it+1
      a(i)=ad(it)
100   continue
      it=0
      in=m+1
      ik=m+max
      do 101 i=in,ik
      it=it+1
      b(i)=bd(it)
101   continue
      i=0
      nt=n+max
      mt=m+max
10    continue
      if(a(nt).ge.b(mt)) goto 1
      np=nt-1
3     continue
      if(np.eq.0) goto 201
      if(a(np).ne.0) goto 2
      np=np-1
```

```
      goto 3
2     continue
      a(np)=a(np)-1
7     continue
      np=np+1
      if(np.eq.nt) goto 30
      a(np)=9
      goto 7
30    continue
      a(nt)=a(nt)+10
      goto 10
1     continue
      i=i+1
      c(i)=a(nt)-b(mt)
      nt=nt-1
      mt=mt-1
      if(mt.ne.0) goto 10
      if(n.eq.m) goto 781
561   continue
      i=i+1
      c(i)=a(nt)
      nt=nt-1
      if(nt.ge.1) goto 561
781   continue
c     write(5,*) 'целая часть разности'
      in=i
      ik=max+1
c     write(5,445) (c(j),j=in,ik,-1)
c     write(5,*) 'дробная часть разности'
      in=max
c     write(5,445) (c(j),j=in,1,-1)
      goto 999
201   continue
      write(5,*) 'из меньшего вычитаем большее,здесь нельзя!'
      stop
999   continue
c     close(5,status='keep')
      kcra=i-max
      kdra=max
      it=0
      in=i
      ik=max+1
      do 2001 j=i,ik,-1
      it=it+1
      cra(it)=c(j)
2001  continue
      im=0
      do 3121 j=1,it
      if(cra(j).ne.0) goto 3122
      im=im+1

3121  continue
      goto 973
3122  continue
      il=0
      do 3123 jp=j,it
      il=il+1
      cra(il)=cra(jp)
3123  continue
973   continue
      kcra=kcra-im
      it=0
      in=max
```

```
      do 2002 j=max,1,-1
      it=it+1
      dra(it)=c(j)
2002  continue
      if(kcra.ne.0) goto 971
      kcra=1
      cra(1)=0
971   continue
      if(kdra.ne.0) goto 972
      kdra=1
      dra(1)=0
972   continue
      if(kdra.gt.krazr) kdra=krazr
c     if(kcra.le.krazr) goto 67291
c     write(*,67292) kcra,krazr
67292 format(1x,'вычитание,количество разрядов в целой части=',i3,
     *' больше  заданного=',i3)
c     pause
c     stop
67291 continue

      return
      end
```

Набор wiczn.for

```
$large
      subroutine wiczn(za,zb,zc,n,nd,a,ad,m,md,b,bd,
     *kcel,kdro,rezc,rezdr,krazr)
c-длинное вычитание с учетом знаков уменьшаемого и вычитаемого
      dimension a(400),b(400),c(400),ad(400),bd(400),cd(400)
      dimension rezc(400),rezdr(400),ar(400),adr(400)
      dimension ai(400),adi(400),bi(400),bdi(400)
      integer a,b,c,rez,os,p,ad,bd,cd,rezc,rezdr,za,zb,zc,zbi
      integer ai,adi,bi,bdi,zais,zbis,ar,adr
      zais=za
      zbis=zb
      do 5671 i=1,n
      ai(i)=a(i)
5671  continue
      do 5672 i=1,nd
      adi(i)=ad(i)
5672  continue
      do 5673 i=1,m
      bi(i)=b(i)
5673  continue
      do 5674 i=1,md
      bdi(i)=bd(i)
5674  continue
      ni=n
      ndi=nd
      mi=m
      mdi=md

c     open(5,file='dlrazzn',status='Unknown')
c     write(*,2001)
2001  format(1x,'введите знак уменьшаемого,1-плюс,-1-минус')
c     read(*,*) za
c     write(*,2002)
2002  format(1x,'введите знак вычитаемого,1-плюс,-1-минус')
c     read(*,*) zb
      zbi=zb
      zb=zb*(-1)
c     write(*,111)
111   format(1x,'введите длину целой части уменьшаемого')
c     read(*,*) n
c     write(*,112)
112   format(1x,'введите длину дробной части уменьшаемого')
c     read(*,*) nd
c     write(*,1111)
1111   format(1x,'введите длину целой части вычитаемого')
c     read(*,*) m
c     write(*,1112)
1112   format(1x,'введите длину дробной части вычитаемого')
c     read(*,*) md


c      write(*,334)
334   format(1x,'введите через пробел разряды целой части',
     *' уменьшаемого')
c     read(*,*) (a(i),i=1,n)
c     write(*,335)
335   format(1x,'введите через пробел разряды дробной части',
     *' уменьшаемого')

c     read(*,*) (ad(i),i=1,nd)
c     write(*,1334)
```

```
1334   format(1x,'введите через пробел разряды целой части',
     *' вычитаемого')

c     read(*,*) (b(i),i=1,m)
c     write(*,1335)
1335   format(1x,'введите через пробел разряды дробной части',
     *' вычитаемого')
c     read(*,*) (bd(i),i=1,md)
c     write(5,*) 'целая часть уменьшаемого'
c     write(5,445) (a(i),i=1,n)
445   format(1x,70i1)
c     write(5,*) 'дробная часть уменьшаемого'
c     write(5,445) (ad(i),i=1,nd)
c     write(5,*) 'целая часть вычитаемого'
c     write(5,445) (b(i),i=1,m)
c     write(5,*) 'дробная часть вычитаемого'
c     write(5,445) (bd(i),i=1,md)
      ip=za*zb
      if(ip.gt.0) goto 781
      if(za.eq.1) goto 782
1901  if(m.gt.n) goto 783
      if(m.eq.n) goto 784
      call wich(n,nd,a,ad,m,md,b,bd,kcel,kdro,rezc,rezdr,krazr)
      zc=za
      goto 785
784   continue
      do 786 i=1,m
      if(a(i).gt.b(i)) goto 787
      if(a(i).lt.b(i)) goto 788
786   continue
      nmin=nd
      if(nmin.gt.md) nmin=md
      do 791 i=1,nmin
      if(ad(i).gt.bd(i)) goto 792
      if(ad(i).lt.bd(i)) goto 793
791   continue
      if(nd.le.md) goto 795
      call wich(n,nd,a,ad,m,md,b,bd,kcel,kdro,rezc,rezdr,krazr)
      zc=za
      goto 785
795   continue
      call wich(m,md,b,bd,n,nd,a,ad,kcel,kdro,rezc,rezdr,krazr)
      zc=zb
      goto 785
793   continue
      call wich(m,md,b,bd,n,nd,a,ad,kcel,kdro,rezc,rezdr,krazr)
      zc=zb
      goto 785
792   continue
      call wich(n,nd,a,ad,m,md,b,bd,kcel,kdro,rezc,rezdr,krazr)
      zc=za
      goto 785
788   continue
      call wich(m,md,b,bd,n,nd,a,ad,kcel,kdro,rezc,rezdr,krazr)
      zc=zb
      goto 785
787   continue
      call wich(n,nd,a,ad,m,md,b,bd,kcel,kdro,rezc,rezdr,krazr)
      zc=za
      goto 785
783   continue
      call wich(m,md,b,bd,n,nd,a,ad,kcel,kdro,rezc,rezdr,krazr)
      zc=zb
```

```
      goto 785
782   continue
      do 1782 i=1,n
      ar(i)=a(i)
1782  continue
      do 1783 i=1,nd
      adr(i)=ad(i)
1783  continue
      nr=n
      ndr=nd
      do 1784 i=1,m
      a(i)=b(i)
1784  continue
      do 1785 i=1,md
      ad(i)=bd(i)
1785  continue
      n=m
      nd=md
      m=nr
      md=ndr
      do 1786 i=1,nr
      b(i)=ar(i)
1786  continue
      do 1787 i=1,ndr
      bd(i)=adr(i)
1787  continue
      zc=za
      za=zb
      zb=zc
      goto 1901
781   continue
      call sl(n,nd,a,ad,m,md,b,bd,kcel,kdro,rezc,rezdr,krazr)
      zc=za
785   continue
c     write(5,901) zc
901   format(1x,'знак разности=',i2)
      zb=zbi
      n=ni
      nd=ndi
      m=mi
      md=mdi
      do 8721 i=1,n
      a(i)=ai(i)
8721  continue
      do 8722 i=1,nd
      ad(i)=adi(i)
8722  continue
      do 8723 i=1,m
      b(i)=bi(i)
8723  continue
      do 8724 i=1,md
      bd(i)=bdi(i)
8724  continue
      za=zais
      zb=zbis
      return
      end
```

Набор ymnzn.for

```
$large
      subroutine ymnzn(za,zb,zc,n,nd,a,ad,m,md,b,bd,
     *kcel,kdro,ctot,cdtot,krazr)
c-длинное умножение с учетом знаков сомножителей
      dimension a(400),b(400),c(400),ad(400),bd(400),cd(400)
      dimension cdtot(400),ctot(400),csl(400),csld(400)
      dimension rezc(400),rezdr(400)
      dimension ai(400),adi(400),bi(400),bdi(400)
      integer a,b,c,rez,os,p,ad,bd,cd,ctot,cdtot,pr,csl,csld
      integer rezc,rezdr,za,zb,zc,ai,adi,bi,bdi,zais,zbis
      zais=za
      zbis=zb
      do 5671 i=1,n
      ai(i)=a(i)
5671  continue
      do 5672 i=1,nd
      adi(i)=ad(i)
5672  continue
      do 5673 i=1,m
      bi(i)=b(i)
5673  continue
      do 5674 i=1,md
      bdi(i)=bd(i)
5674  continue
      ni=n
      ndi=nd
      mi=m
      mdi=md

      zc=za*zb
c     open(5,file='dlinymno',status='Unknown')
c     write(*,111)
111   format(1x,'введите длину целой части первого множителя')
c     read(*,*) n
c     write(*,112)
112   format(1x,'введите длину дробной части первого множителя')
c     read(*,*) nd
c     write(*,1111)
1111   format(1x,'введите длину целой части второго множителя')
c     read(*,*) m
c     write(*,1112)
1112   format(1x,'введите длину дробной части второго множителя')
c     read(*,*) md


c      write(*,334)
334   format(1x,'введите через пробел разряды целой части',
     *' первого множителя')
c     read(*,*) (a(i),i=1,n)
c     write(*,335)
335   format(1x,'введите через пробел разряды дробной части',
     *' первого множителя')

c     read(*,*) (ad(i),i=1,nd)
c     write(*,1334)
1334   format(1x,'введите через пробел разряды целой части',
     *' второго множителя')

c     read(*,*) (b(i),i=1,m)
c     write(*,1335)
1335   format(1x,'введите через пробел разряды дробной части',
```

```
     *' второго множителя')


c     read(*,*) (bd(i),i=1,md)
c     write(5,*) 'целая часть первого множителя'
c     write(5,445) (a(i),i=1,n)
445   format(1x,70i1)
c     write(5,*) 'дробная часть первого множителя'
c     write(5,445) (ad(i),i=1,nd)
c     write(5,*) 'целая часть второго множителя'
c     write(5,445) (b(i),i=1,m)
c     write(5,*) 'дробная часть второго множителя'
c     write(5,445) (bd(i),i=1,md)

      call ymn(n,nd,a,ad,m,md,b,bd,kcel,kdro,ctot,cdtot,krazr)

c     write(5,*) 'целая часть произведения'
c     write(5,445) (ctot(i),i=1,kcel)
c     write(5,*) 'дробная часть произведения'
c     write(5,445) (cdtot(i),i=1,kdro)
c     close(5,status='keep')
      n=ni
      nd=ndi
      m=mi
      md=mdi
      do 8721 i=1,n
      a(i)=ai(i)
8721  continue
      do 8722 i=1,nd
      ad(i)=adi(i)
8722  continue
      do 8723 i=1,m
      b(i)=bi(i)
8723  continue
      do 8724 i=1,md
      bd(i)=bdi(i)
8724  continue
      za=zais
      zb=zbis
      return
      end
```

c-ymn

```
c-длинное умножение
$large
      subroutine ymn(n,nd,a,ad,m,md,b,bd,kcel,kdro,ctot,cdtot,krazr)
      dimension a(400),b(400),c(400),ad(400),bd(400),cd(400)
      dimension cdtot(400),ctot(400),csl(400),csld(400)
      dimension rezc(400),rezdr(400)
      integer a,b,c,rez,os,p,ad,bd,cd,ctot,cdtot,pr,csl,csld
      integer rezc,rezdr
c     open(5,file='ymno',status='Unknown')
c     write(*,111)
111   format(1x,'введите длину целой части первого множителя')
c     read(*,*) n
c     write(*,112)
112   format(1x,'введите длину дробной части первого множителя')
c     read(*,*) nd
c     write(*,1111)
1111   format(1x,'введите длину целой части второго множителя')
c     read(*,*) m
c     write(*,1112)
1112   format(1x,'введите длину дробной части второго множителя')
```

```
c     read(*,*) md


c      write(*,334)
334   format(1x,'введите через пробел разряды целой части',
     *' первого множителя')
c     read(*,*) (a(i),i=1,n)
c     write(*,335)
335   format(1x,'введите через пробел разряды дробной части',
     *' первого множителя')

c     read(*,*) (ad(i),i=1,nd)
c     write(*,1334)
1334   format(1x,'введите через пробел разряды целой части',
     *' второго множителя')

c     read(*,*) (b(i),i=1,m)
c     write(*,1335)
1335   format(1x,'введите через пробел разряды дробной части',
     *' второго множителя')


c     read(*,*) (bd(i),i=1,md)
c     write(5,*) 'целая часть первого множителя'
c     write(5,445) (a(i),i=1,n)
445   format(1x,70i1)
c     write(5,*) 'дробная часть первого множителя'
c     write(5,445) (ad(i),i=1,nd)
c     write(5,*) 'целая часть второго множителя'
c     write(5,445) (b(i),i=1,m)
c     write(5,*) 'дробная часть второго множителя'
c     write(5,445) (bd(i),i=1,md)
c     if(krazr.le.100) goto 66791
c     write(*,*) 'умножение,количество разрядов>100!'
c     pause
c     stop
66791 continue
      do 66987 i=1,400
      ctot(i)=0
66987 cdtot(i)=0
c-формирование двух массивов для умножения
      it=0
      in=n+1
      ik=n+nd
      do 100 i=in,ik
      it=it+1
      a(i)=ad(it)
100   continue
      it=0
      in=m+1
      ik=m+md
      do 101 i=in,ik
      it=it+1
      b(i)=bd(it)
101   continue
      ntot=n+m
      kdtot=nd+md
      do 651 i=1,ntot
      ctot(i)=0
651   continue
      do 652 i=1,kdtot
      cdtot(i)=0
652   continue
```

```
c-умножение
      i=0
      im=0
      p=0
      nt=n+nd
      mt=m+md
801   continue
      p=0
      do 941 jp=1,400
      c(jp)=0
941   continue
      i=0
      nt=n+nd
      im=im+1
701   continue
      pr=a(nt)*b(mt)+p
      i1=pr/10
      os=pr-i1*10
      i=i+1
      c(i)=os
      p=i1

      nt=nt-1
      if(nt.ge.1) goto 701
      if(p.eq.0) goto 1701
      i=i+1
      c(i)=p
      p=0
1701  continue
c     write(5,942) im,i,(c(jp),jp=1,i)
942   format(1x,'im=',i3,'i dlj c=',i3/1x,79i1)
      kraz=md-im+1
      kcd=nd+kraz
      icel=i-kcd
      if(icel.lt.i) goto 1921
      iab=abs(kcd)
      it=0
      do 1922 jp=i,1,-1
      it=it+1
      csl(it)=c(jp)
1922  continue
      if(iab.eq.0) goto 1941
      do 1923 jp=iab,1,-1
      it=it+1
      csl(it)=0
1923  continue
1941  continue
      idr=1
      csld(1)=0
      goto 8722
1921  continue

      it=0
      in=i
      ik=kcd+1
      if(in.eq.ik) goto 961
      if(in.lt.ik) goto 962
      do 702 j=in,ik,-1
      it=it+1
      csl(it)=c(j)
702   continue
      icel=i-kcd
```

```
      goto 963
961   continue
      icel=1
      csl(1)=c(in)
      goto 963
962   continue
      icel=1
      csl(1)=0
963   continue
      in=kcd
      it=0
      do 703 j=in,1,-1
      it=it+1
      csld(it)=c(j)
703   continue
      idr=kcd
      if(idr.gt.0) goto 8722
      idr=1
      csld(1)=0
8722  continue
c     write(5,*) 'целая часть первого слагаемого'
c     write(5,445) (csl(j),j=1,icel)
c     write(5,*) 'дробная часть первого слагаемого'
c     write(5,445) (csld(j),j=1,idr)
c     write(5,*) 'целая часть второго слагаемого'
c     write(5,445) (ctot(j),j=1,ntot)
c     write(5,*) 'дробная часть второго слагаемого'
c     write(5,445) (cdtot(j),j=1,kdtot)

      call sl(icel,idr,csl,csld,ntot,kdtot,ctot,cdtot,kcel,kdro,
     *rezc,rezdr,krazr)
      if(kcel.ne.0) goto 8723
      kcel=1
      rezc(1)=0
8723  continue
      if(kdro.ne.0) goto 8724
      kdro=1
      rezdr(1)=0
8724  continue
c     write(5,*) 'целая часть суммы'
c     write(5,445) (rezc(j),j=1,kcel)
c     write(5,*) 'дробная часть суммы'
c     write(5,445) (rezdr(j),j=1,kdro)

      do 704 i=1,kcel
      ctot(i)=rezc(i)
704   continue
      do 705 i=1,kdro
      cdtot(i)=rezdr(i)
705   continue
      ntot=kcel
      kdtot=kdro
      mt=mt-1
      if(mt.ge.1) goto 801
c     write(5,*) 'целая часть произведения'
c     write(5,445) (ctot(i),i=1,kcel)
c     write(5,*) 'дробная часть произведения'
c     write(5,445) (cdtot(i),i=1,kdro)
c     close(5,status='keep')
      it=0
      do 7766 j=1,kcel
      if(ctot(j).ne.0) goto 7767
      it=it+1
```

```
7766  continue
7767  continue
      il=0
      do 7768 jp=j,kcel
      il=il+1
      ctot(il)=ctot(jp)
7768  continue
      kcel=kcel-it
      if(kcel.ne.0) goto 3999
      kcel=1
      ctot(1)=0
3999  continue
      if(kdro.ne.0) goto 4000
      kdro=1
      cdtot(1)=0
4000  continue

      if(kdro.gt.krazr) kdro=krazr
c     if(kcel.le.krazr) goto 67291
c     write(*,67292) kcel,krazr
67292 format(1x,'умножение,количество разрядов в целой части=',i3,
     *' больше  заданного=',i3)
c     pause
c     stop
67291 continue

      return
      end
```

Набор delzn.for

```
c-длинное деление с учетом знаков
$large
      subroutine delzn(za,zb,zc,n,nd,a,ad,m,md,b,bd,
     *kcdel,kddel,cdel,ddel,krazr)
      dimension a(400),ad(400),b(400),bd(400),c(400),cd(400)
      dimension ar(400),adr(400),br(400),bdr(400)
      dimension cdel(400),ddel(400),ctot(400),cdtot(400)
      dimension rezc(400),rezdr(400),ce(400),ced(400)
      integer a,ad,b,bd,c,cd,cdel,ddel,ctot,cdtot,rezc,rezdr
      integer ce,ced,za,zb,zc,ar,adr,br,bdr,zais,zbis
      zais=za
      zbis=zb
      do 5671 i=1,n
      ar(i)=a(i)
5671  continue
      do 5672 i=1,nd
      adr(i)=ad(i)
5672  continue
      do 5673 i=1,m
      br(i)=b(i)
5673  continue
      do 5674 i=1,md
      bdr(i)=bd(i)
5674  continue
      ni=n
      ndi=nd
      mi=m
      mdi=md
      zc=za*zb
c     open(5,file='dlindel',status='Unknown')
c      write(*,111)
111   format(1x,'введите длину целой части делимого')
c     read(*,*) n
c     write(*,112)
112   format(1x,'введите длину дробной части делимого')
c     read(*,*) nd
c     write(*,1111)
1111   format(1x,'введите длину целой части делителя')
c     read(*,*) m
c     write(*,1112)
1112   format(1x,'введите длину дробной части делителя')
c     read(*,*) md


c      write(*,334)
334   format(1x,'введите через пробел разряды целой части',
     *' делимого')
c     read(*,*) (a(i),i=1,n)
c     write(*,335)
335   format(1x,'введите через пробел разряды дробной части',
     *' делимого')

c     read(*,*) (ad(i),i=1,nd)
c     write(*,1334)
1334   format(1x,'введите через пробел разряды целой части',
     *' делителя')

c     read(*,*) (b(i),i=1,m)
c     write(*,1335)
1335   format(1x,'введите через пробел разряды дробной части',
     *' делителя')
```

```
c     read(*,*) (bd(i),i=1,md)
c     write(5,*) 'целая часть делимого'
c     write(5,445) (a(i),i=1,n)
445   format(1x,79i1)
c     write(5,*) 'дробная часть делимого'
c     write(5,445) (ad(i),i=1,nd)
c     write(5,*) 'целая часть делителя'
c     write(5,445) (b(i),i=1,m)
c     write(5,*) 'дробная часть делителя'
c     write(5,445) (bd(i),i=1,md)

      call deld(n,nd,a,ad,m,md,b,bd,kcdel,kddel,cdel,ddel,krazr)

c        write(5,*) 'целая часть частного'
c     write(5,445) (cdel(j),j=1,kcdel)
c     write(5,*) 'дробная часть частного'
c     write(5,445) (ddel(j),j=1,kddel)
c     close(5,status='keep')
      n=ni
      nd=ndi
      m=mi
      md=mdi
      do 8721 i=1,n
      a(i)=ar(i)
8721  continue
      do 8722 i=1,nd
      ad(i)=adr(i)
8722  continue
      do 8723 i=1,m
      b(i)=br(i)
8723  continue
      do 8724 i=1,md
      bd(i)=bdr(i)
8724  continue
      za=zais
      zb=zbis
      return
      end
```

c-deld

```
$large
      subroutine deld(n,nd,a,ad,m,md,b,bd,kcdel,kddel,cdel,ddel,krazr)
c-длинное деление
      dimension a(400),ad(400),b(400),bd(400),c(400),cd(400)
      dimension cdel(400),ddel(400),ctot(400),cdtot(400)
      dimension rezc(400),rezdr(400),ce(400),ced(400)
      integer a,ad,b,bd,c,cd,cdel,ddel,ctot,cdtot,rezc,rezdr
      integer ce,ced
c     if(krazr.le.100) goto 66791
c     write(*,*) 'деление,количество разрядов>100!'
c     pause
c     stop
66791 continue
      do 66789 i=1,400
      cdel(i)=0
66789 ddel(i)=0

c-формирование двух массивов для деления
      max=nd
      if(max.lt.md) max=md
      if(md.eq.max) goto 78
```

```
      in=md+1
      do 79 i=in,max
      bd(i)=0
79    continue
      goto 81
78    continue
      in=nd+1
      do 82 i=in,max
      ad(i)=0
82    continue
81    continue
      it=0
      in=n+1
      ik=n+max
      do 100 i=in,ik
      it=it+1
      a(i)=ad(it)
100   continue
      il=0
      do 5671 j=1,ik
      if(a(j).ne.0) goto 5672
      il=il+1
5671  continue
      kcdel=0
      kddel=0
      goto 1231
c     stop
5672  continue
      it=0
      do 5673 jp=j,ik
      it=it+1
      a(it)=a(jp)
5673  continue
      it=0
      in=m+1
      ik=m+max
      do 101 i=in,ik
      it=it+1
      b(i)=bd(it)
101   continue
      il1=0
      do 15671 j=1,ik
      if(b(j).ne.0) goto 15672
      il1=il1+1
15671 continue
      kcdel=0
      kddel=0
c     write(*,*) 'на ноль делить нельзя'
      goto 1231
c     stop
15672 continue
      it=0
      do 15673 jp=j,ik
      it=it+1
      b(it)=b(jp)
15673 continue

c        if(n.lt.m) goto 675
c     if(n.eq.m) goto 676
      do 2677 i=1,400
      ctot(i)=0
      cdtot(i)=0
2677  continue
```

```
      kctot=1
      kdto=1
      ce(1)=1
      kedi=1
      ced(1)=0
      kedid=1
      kad=1
      kbd=1
      ad(1)=0
      bd(1)=0
      nat=n+max-il
      mat=m+max-il1
      isd=0
      kcel=1
      kdro=1
      rezc(1)=0
      rezdr(1)=0
2010    continue
c     write(5,4501) nat,mat
4501  format(1x,'nat=',i3,' mat=',i3)
      if(nat.lt.mat) goto 675
      if(nat.eq.mat) goto 676
      goto 677
675     continue
      if(isd.ne.0) goto 5621
      do 5701 j=1,kcel
      cdel(j)=rezc(j)
5701    continue
      do 5702 j=1,kdro
      ddel(j)=rezdr(j)
5702  continue
      rezc(1)=0
      kcdel=kcel
      kddel=kdro
      if(nat.eq.0) goto 1231
5621  continue
      if(isd.eq.0) goto 8721
c     write(5,3451) isd,rezc(1)
3451  format(1x,'isd=',i3,' rezc(1)=',i3)
      ddel(isd)=rezc(1)
      rezc(1)=0
      kddel=isd
8721  continue
c     nat=nat+1
c     isd=isd+1
c     kddel=isd
      if(nat.eq.0) goto 1231
      do 9781 j=1,nat
c     write(5,9788) nat,isd,j,a(j)
9788  format(1x,'nat=',i3,' isd=',i3,' j=',i3,' a(j)=',i3)
      if(a(j).ne.0) goto 9782
9781  continue
      goto 1231
9782  continue
      nat=nat+1
      isd=isd+1
      if(isd.gt.krazr) goto 1231
      a(nat)=0
      do 1677 i=1,400
      ctot(i)=0
      cdtot(i)=0
1677  continue
      kctot=1
```

```
      kdto=1
      goto 2010
676   continue
      do 4562 j=1,nat
      if(a(j).gt.b(j)) goto 677
      if(a(j).lt.b(j)) goto 675
4562  continue
677   continue
      call wich(nat,kad,a,ad,mat,kbd,b,bd,kc,kcd,c,cd,krazr)
      call sl(kctot,kdto,ctot,cdtot,kedi,kedid,ce,ced,
     *kcel,kdro,rezc,rezdr,krazr)
      kctot=kcel
      kdto=kdro
      do 1121 i=1,kcel
      ctot(i)=rezc(i)
1121  continue
      do 1122 i=1,kdro
      cdtot(i)=rezdr(i)
1122    continue
      nat=kc
      kad=kcd
      if(nat.eq.0) a(1)=0
      do 3001 j=1,nat
c     if(nat.ne.0) goto 3112
c     write(5,3113) nat,j,a(j),c(j)
3113  format(1x,'nat=',i3,' j=',i3,' a(j)=',i3,' c(j)=',i3)
3112  continue
      a(j)=c(j)
3001  continue
      do 3002 j=1,kad
      ad(j)=cd(j)
3002  continue
      goto 2010
1231  continue
      if(kcdel.ne.0) goto 971
      kcdel=1
      cdel(1)=0
971   continue
      if(kddel.ne.0) goto 972
      kddel=1
      ddel(1)=0
972   continue
      if(kddel.gt.krazr) kddel=krazr
c     if(kcdel.le.krazr) goto 67291
c     write(*,67292) kcdel,krazr
67292 format(1x,'деление,количество разрядов в целой части=',i3,
     *' больше  заданного=',i3)
c     pause
c     stop
67291 continue

c     write(5,*) 'целая часть частного'
c     write(5,445) (cdel(j),j=1,kcdel)
c     write(5,*) 'дробная часть частного'
c     write(5,445) (ddel(j),j=1,kddel)
c     close(5,status='keep')
      return
      end
```

ПРИЛОЖЕНИЕ-2(наборы с исходными текстами программ на C)
Набор pidlpar.c(вычисление числа пи с имитацией распараллеливания процедуры суммирования)

```
//#include<windows.h>
#include<string.h>
#include<stdio.h>
//#include"mnkcpp.h"
#include<math.h>
//#include<alloc.h>
//#include"mpi.h"
#include<math.h>
int main( int argc, char *argv[ ] )
      {
FILE *fp;
extern void delzn(int izr,int iznt,int *pizrez,int ncr,
    int ndr,int *rc,int *rd,
    int nct,int ndt,int *tc,int *td,
     int *pkcrez,int *pkdrez,int *rezc,int *rezd,int krazr);

     extern void ymnzn(int izp,int izr,int *pizrez,int ncp,int ndp,
int *pc,int *pd,
int ncr,int ndr,int *rc,int *rd,
     int *pkcrez,int *pkdrez,int *rezc,int *rezd,int krazr);

extern void wiczn(int izr1,int izr,int *pizrez,int ncr1,int ndr1,
int *rc1,int *rd1,
int ncr,int ndr,int *rc,int *rd,
int *pkcrez,int *pkdrez,int *rezc,int *rezd,int krazr);

extern void slzn(int izsum,int izn2,int *pizrez,int isc,int isd,
int *icsum,int *idsum,
int ikc2,int ikd2,
     int *imc2,int *imd2,
int *pkcrez,int *pkdrez,int *icrez,int *idrez,int krazr);


 int myid, numprocs,izkop,kckop,kdkop,l;
//    MPI_Status status;

// MPI_Init(&argc, &argv);
//MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
//MPI_Comm_rank(MPI_COMM_WORLD,&myid);

      int ixc[401],ixd[401],isuc[401],isud[401],idxc[401],
      idxd[401],icrez[401],idrez[401],icr1[401],ios,irab[401],
      idr1[401],icr2[401],idr2[401],iced[401],ided[401],ikp,
      ic4[401],id4[401],icpr[401],idpr[401],ickop[401],
      idkop[401],idxn[401],icxn[401];
      int izrez,ic1,id1,iznak,ic2,id2,kcrez,kdrez,iz1,
      izdel,izymn,izsum;
      int izn1,kced,kded,izndx,kcd,i,izn4,kc4,kd4,iznx,kcx,
      kdx,iznax,kcxn,kdxn,iznsu,kcsu,kdsu,isch,iznpr,kcpr,
      kdpr,kdd,krazr,kproc;
      int izr,iznt,*pizrez, ncr,
    ndr,rc[401], rd[401],
    nct,ndt,tc[401],td[401],
     *pkcrez,*pkdrez, rezc[401], rezd[401];
      int izp,    ncp,  ndp,
  pc[401],  pd[401];
int izr1,   ncr1,   ndr1,
   rc1[401],   rd1[401];
```

```
int    izn2,    isc,   isd,
   icsum[401],   idsum[401],
   ikc2,   ikd2,
        imc2[401],   imd2[401];
//    icrez[401],   idrez[401];

         fp=fopen("pidlp_c","w");
//    open(6,file='pipardl',status='Unknown')
/* c-kcx-количество разрядов в целой части x
c-kdx-количество разрядов в дробной части x
c-kcsu-количество разрядов в целой части интегральной суммы
c-kdsu-количество разрядов в дробной части интегральной суммы
c-ixc-массив разрядов целой части x
c-ixd-массив разрядов дробной части x
c-isuc-массив разрядов целой части интегральной суммы
c-isud-массив разрядов дробной части интегральной суммы
c-idxc-массив разрядов целой части dx
c-idxd-массив разрядов дробной части dx
c-kcd-количество разрядов в целой части dx
c-kdd-количество разрядов в дробной части dx
c-iced-массив разрядов целой части 1(единицы)
c-ided-массив разрядов дробной части 1(единицы)
c-kced-количество разрядов в целой части 1(единицы)
c-kded-количество разрядов в дробной части 1(единицы) */
      printf("введите количество разрядов в шаге dx\n");
      scanf("%d",&kdd);
//     kdd=6;

      printf("\nвведите количество разрядов в дробной части\n");
      scanf("%d",&krazr);
//     krazr=20;

      printf("\nвведите количество процессов\n");
      scanf("%d",&kproc);
//     kproc=numprocs;

//    write(6,873) kdd,krazr,kproc
   fprintf(fp,"количество разрядов в шаге dx=%d\n",kdd);
   fprintf(fp,"количество разрядов в дробной части=%d\n",krazr);
   fprintf(fp,"количество процессов=%d\n",kproc);
//c   krazr=10
//c-количество процессов
      izkop=1;
      kdkop=1;
//c-формирование массива целой части количества процессов
      ikp=0;
m2001:;
      ios=kproc-kproc/10*10;
      ikp=ikp+1;
      irab[ikp]=ios;
      kproc=kproc/10;
      if(kproc!=0) goto m2001;
      kckop=ikp;
      l=0;
      for(i=ikp;i>=1;i=i-1)
      {
      l=l+1;
      ickop[l]=irab[i];
      }

      idkop[1]=0;

//c-определяем 1(единицу)
```

```
       izn1=1;
       kced=1;
       kded=1;
      iced[1]=1;
      ided[1]=0;
//c- определяем dx
       izndx=1;
       kcd=1;
//c   kdd=2
      idxc[1]=0;
//    do 1 i=1,kdd
      for(i=1;i<=kdd;i++)

      idxd[i]=0;
      idxd[kdd]=1;
//c-определяем 4(четыре)
       izn4=1;
       kc4=1;
       kd4=1;
      ic4[1]=4;
      id4[1]=0;
//c-определяем начальное x(x=0)
       iznx=1;
       kcx=1;
       kdx=1;
      ixc[1]=0;
      ixd[1]=0;
//c-меняющееся начальное x
       iznax=1;
       kcxn=1;
       kdxn=1;
      icxn[1]=0;
      idxn[1]=0;
//c-определяем начальное значение интегральной суммы
       iznsu=1;
       kcsu=1;
       kdsu=1;
      isuc[1]=0;
      isud[1]=0;
       isch=0;
//c-определяем номер процесса
       iznpr=1;
       kcpr=1;
       kdpr=1;
      icpr[1]=1;
      idpr[1]=0;
m777:;
//c-вычитаем из номера процесса единицу
       wiczn(iznpr,izn1,&izrez,
     kcpr,kdpr,icpr,idpr,
     kced,kded,iced,ided,
     &ic1,&id1,icr1,idr1,krazr);
//c-умножаем (npr-1)*dx
      ymnzn(izrez,izndx,&iznak,
     ic1,id1,icr1,idr1,
     kcd,kdd,idxc,idxd,
     &ic2,&id2,icr2,idr2,krazr);
//c-складываем x+(npr-1)*dx
      slzn(iznax,iznak,&izrez,
     kcxn,kdxn,icxn,idxn,
     ic2,id2,icr2,idr2,
     &ic1,&id1,icr1,idr1,krazr);
//c-переопределение x
```

```
      iznx=izrez;
      kcx=ic1;
      kdx=id1;
//    do 210 i=1,kcx
      for(i=1;i<=kcx;i++)

      ixc[i]=icr1[i];
      for(i=1;i<=kdx;i++)
      ixd[i]=idr1[i];
      isch=isch+1;
      if(isch<1000) goto m678;
      if(ixc[1]>=iced[1]) goto m1111;
//c   write(6,891) icpr(1)
       printf("номер процесса\n");
       for(i=1;i<=kcpr;i++)
       printf("%1d",icpr[i]);
       printf("\n");
//    write(*,*) 'дробная часть x'
//    write(*,34) (ixd(l),l=1,kdx)
      isch=0;
m678:;
//c-сравниваем x с единицей
      if(ixc[1]>=iced[1]) goto m1111;

//c-умножаем x на x
       ymnzn(iznx,iznx,&izrez,
     kcx,kdx,ixc,ixd,
     kcx,kdx,ixc,ixd,
     &kcrez,&kdrez,icrez,idrez,krazr);
//c-складываем 1+x*x
       slzn(izn1,izrez,&iz1,
     kced,kded,iced,ided,
     kcrez,kdrez,icrez,idrez,
     &ic1,&id1,icr1,idr1,krazr);
//c-делим 1/(1+x*x)
       delzn(izn1,iz1,&izdel,
     kced,kded,iced,ided,
     ic1,id1,icr1,idr1,
     &ic2,&id2,icr2,idr2,krazr);
//c-умножаем 1/(1+x*x) на dx
       ymnzn(izdel,izndx,&izymn,
     ic2,id2,icr2,idr2,
     kcd,kdd,idxc,idxd,
     &ic1,&id1,icr1,idr1,krazr);
//c-наращиваем интегральную сумму
       slzn(iznsu,izymn,&izsum,
     kcsu,kdsu,isuc,isud,
     ic1,id1,icr1,idr1,
     &ic2,&id2,icr2,idr2,krazr);
//c-переопределяем интегральную сумму
       iznsu=izsum;
       kcsu=ic2;
       kdsu=id2;
       for(i=1;i<=ic2;i++)
       isuc[i]=icr2[i];
       for(i=1;i<=id2;i++)
       isud[i]=idr2[i];
//c-увеличить меняющееся начальное x на kproc*dx
//c-вычислить kproc*dx
       ymnzn(izkop,izndx,&izrez,
     kckop,kdkop,ickop,idkop,
     kcd,kdd,idxc,idxd,
     &ic1,&id1,icr1,idr1,krazr);
```

```
//c-суммируем меняющееся начальное x+kproc*dx
      slzn(iznax,izrez,&iz1,
     kcxn,kdxn,icxn,idxn,
     ic1,id1,icr1,idr1,
     &ic2,&id2,icr2,idr2,krazr);
//c-переопределяем меняющееся начальное x
      iznax=iz1;
      kcxn=ic2;
      kdxn=id2;
      for(i=1;i<=kcxn;i++)
      icxn[i]=icr2[i];
      for( i=1;i<=kdxn;i++)
      idxn[i]=idr2[i];
      goto m777;
m1111:;

//c-увеличиваем номер процесса на один
      slzn(iznpr,izn1,&izrez,
     kcpr,kdpr,icpr,idpr,
     kced,kded,iced,ided,
     &ic1,&id1,icr1,idr1,krazr);
//c-переопределяем номер процесса
      iznpr=izrez;
      kcpr=ic1;
      kdpr=id1;
      for(i=1;i<=kcpr;i++)
      icpr[i]=icr1[i];
      for(i=1;i<=kdpr;i++)
      idpr[i]=idr1[i];
//c-сравнение номера процесса с количеством процессов
//    c-сравнение номера процесса с количеством процессов
      if(kcpr>kckop) goto m11;
      if(kcpr<kckop) goto m4002;
      for(i=1;i<=kcpr;i++)
      if(icpr[i]>ickop[i]) goto m11;
m4002:;
      iznax=1;
      kcxn=1;
      kdxn=1;
      icxn[1]=0;
      idxn[1]=0;
      goto m777;
m11:;
//c-уьножение интегральной суммы на 4
      ymnzn(iznsu,izn4,&izrez,
     kcsu,kdsu,isuc,isud,
     kc4,kd4,ic4,id4,
     &kcrez,&kdrez,icrez,idrez,krazr);
//c-вывод числа пи
//c   open(6,file='ch_pi',status='Unknown')
//    write(6,41)
      fprintf(fp,"шаг интегрирования\n");
//    write(6,42)
      fprintf(fp,"целая часть шага\n");
//    write(6,34)
      for(i=1;i<=kcd;i++)
      fprintf(fp,"%1d",idxc[i]);
      fprintf(fp,"\n");
//    write(6,43)
      fprintf(fp,"дробная часть шага\n");
//    write(6,34)
      for(i=1;i<=kdd;i++)
      fprintf(fp,"%1d",idxd[i]);
```

```
      fprintf(fp,"\n");
//    write(6,31)
      fprintf(fp,"число пи\n");
//    write(6,33)
//    write(*,33)
      fprintf(fp,"целая часть\n");
//    write(6,34) (icrez(i),i=1,kcrez)
      for(i=1;i<=kcrez;i++)
      fprintf(fp,"%1d",icrez[i]);
      fprintf(fp,"\n");
//34  format(/1x,70i1)
       fprintf(fp,"дробная часть\n");
//    write(6,34) (idrez(i),i=1,kdrez)
      for(i=1;i<=kdrez;i++)
      fprintf(fp,"%1d",idrez[i]);
      fprintf(fp,"\n");
      fclose(fp);
      }
```

Набор sl.c

```
#include<stdio.h>
#include<math.h>
//#include<alloc.h>

//c-длинное сложение
      void sl(int n,int nd,int *a,int *ad,
  int m,int md,int *b,int *bd,
int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr)
{
int c[401],cd[401],i,j,ir,max,in,kcel,kdro,
it,ik,p,nt,mt,rez,i1,os;

        for(i=1;i<=399;i++)
      {
      rezc[i]=0;
      rezdr[i]=0;
      }
      if(n>=m) goto m897;
      for(i=1;i<=n;i++)
      c[i]=a[i];
      for(i=1;i<=nd;i++)
      cd[i]=ad[i];
      for(i=1;i<=m;i++)
      a[i]=b[i];
      for(i=1;i<=md;i++)
      ad[i]=bd[i];
      for(i=1;i<=n;i++)
      b[i]=c[i];
      for(i=1;i<=nd;i++)
      bd[i]=cd[i];
      ir=n;
      n=m;
      m=ir;
      ir=nd;
      nd=md;
      md=ir;
m897:;
//c-формирование двух массивов для сложения
      max=nd;
      if(max<md) max=md;
      if(md==max) goto m78;
      in=md+1;
      for(i=in;i<=max;i++)
      bd[i]=0;
      goto m81;
m78:;
      in=nd+1;
      for(i=in;i<=max;i++)
      ad[i]=0;
m81:;
      it=0;
      in=n+1;
      ik=n+max;
      for(i=in;i<=ik;i++)
      {
      it=it+1;
      a[i]=ad[it];
      }
      it=0;
      in=m+1;
      ik=m+max;
```

```
      for(i=in;i<=ik;i++)
      {
      it=it+1;
      b[i]=bd[it];
      }

//c-сложение
      i=0;
      p=0;
      nt=n+max;
      mt=m+max;
m7:;
      if(mt<1) rez=a[nt];
      if(mt>=1) rez=a[nt]+b[mt];
      rez=rez+p;
      i1=rez/10;
      os=rez-i1*10;
      i=i+1;
      c[i]=os;
//c   if(i.ne.1) goto 1
      p=i1;
      nt=nt-1;
      mt=mt-1;
      if((p==0)&&(mt<1)) goto m4;
      if((nt<1)&&(p!=0)) goto m5;
      if((nt<1)&&(p==0)) goto m4;
      goto m7;
m5:;
      i=i+1;
      c[i]=p;
      goto m67;
m4:;

m89:;
      if(nt<1) goto m67;
      i=i+1;
      c[i]=a[nt];
      nt=nt-1;
      goto m89;
m67:;
      ik=max+1;
      kcel=i-max;
      kdro=max;
      it=0;
      for(j=i;j>=ik;j=j-1)
      {
      it=it+1;
      rezc[it]=c[j];
      }
      it=0;
      for(j=max;j>=1;j=j-1)
      {
      it=it+1;
      rezdr[it]=c[j];
      }
      if(kcel!=0) goto m971;
      kcel=1;
      rezc[1]=0;
m971:;
      if(kdro!=0) goto m972;
      kdro=1;
      rezdr[1]=0;
m972:;
```

```
if(kdro>krazr) kdro=krazr;
*pkcel=kcel;
*pkdro=kdro;
return;
}
```

Набор slzn.c

```
#include<stdio.h>
#include<math.h>
//#include<alloc.h>

      void slzn(int za,int zb,int *pzc,
 int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
     int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr)
{

 extern void sl(int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr);

extern void wich(int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr);

//c-длинное сложение с учетом знаков слагаемых

int c[401],cd[401],kcel,kdro,
ar[401],adr[401],zc,
ai[401],adi[401],bi[401],bdi[401];
int zais,zbis,i,ni,ndi,mi,mdi,ip,nmin,nr,ndr;
      zais=za;
      zbis=zb;
      for(i=1;i<=n;i++)
      ai[i]=a[i];
      for(i=1;i<=nd;i++)
      adi[i]=ad[i];
      for(i=1;i<=m;i++)
      bi[i]=b[i];
      for(i=1;i<=md;i++)
      bdi[i]=bd[i];
      ni=n;
      ndi=nd;
      mi=m;
      mdi=md;

      ip=za*zb;
      if(ip>0) goto m781;
      if(za==1) goto m782;
m1901:      if(m>n) goto m783;
      if(m==n) goto m784;
      wich(n,nd,a,ad,m,md,b,bd,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=za;
      goto m785;
m784:;
      for(i=1;i<=m;i++)
      {
      if(a[i]>b[i]) goto m787;
      if(a[i]<b[i]) goto m788;
      }
      nmin=nd;
      if(nmin>md) nmin=md;
      for(i=1;i<=nmin;i++)
      {
      if(ad[i]>bd[i]) goto m792;
      if(ad[i]<bd[i]) goto m793;
      }
      if(nd<=md) goto m795;
```

```
      wich(n,nd,a,ad,m,md,b,bd,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=za;
      goto m785;
m795:;
      wich(m,md,b,bd,n,nd,a,ad,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=zb;
      goto m785;
m793:;
      wich(m,md,b,bd,n,nd,a,ad,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=zb;
      goto m785;
m792:;
      wich(n,nd,a,ad,m,md,b,bd,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=za;
      goto m785;
m788:;
      wich(m,md,b,bd,n,nd,a,ad,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=zb;
      goto m785;
m787:;
      wich(n,nd,a,ad,m,md,b,bd,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=za;
      goto m785;
m783:;
      wich(m,md,b,bd,n,nd,a,ad,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=zb;
      goto m785;
m782:;
      for(i=1;i<=n;i++)
      ar[i]=a[i];

      for(i=1;i<=nd;i++)
      adr[i]=ad[i];
      nr=n;
      ndr=nd;
      for(i=1;i<=m;i++)
      a[i]=b[i];
      for(i=1;i<=md;i++)
      ad[i]=bd[i];
      n=m;
      nd=md;
      m=nr;
      md=ndr;
      for(i=1;i<=nr;i++)
      b[i]=ar[i];
      for(i=1;i<=ndr;i++)
      bd[i]=adr[i];
      zc=za;
      za=zb;
      zb=zc;
      goto m1901;
m781:;
      sl(n,nd,a,ad,m,md,b,bd,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=za;
m785:;
      n=ni;
      nd=ndi;
      m=mi;
      md=mdi;
      for(i=1;i<=n;i++)
      a[i]=ai[i];
      for(i=1;i<=nd;i++)
      ad[i]=adi[i];
```

```
for(i=1;i<=m;i++)
b[i]=bi[i];
for(i=1;i<=md;i++)
bd[i]=bdi[i];

za=zais;
zb=zbis;
*pkcel=kcel;
*pkdro=kdro;
return;
}
```

Набор wich.c

```
#include<stdio.h>
#include<math.h>
//#include<alloc.h>

//c-длинное вычитание
      void wich(int n,int nd,int *a,int *ad,
 int m,int md,int *b,int *bd,
 int *pkcra,int *pkdra,int *cra,int *dra,int krazr)
      {
int c[401],cd[401],i,ir,max,jp,kcra,kdra,
in,it,ik,nt,mt,np,il,j,im;

      for(i=1;i<=399;i++)
      {
      cra[i]=0;
      dra[i]=0;
      }
      if(n>=m) goto m897;
      for(i=1;i<=n;i++)
      c[i]=a[i];
      for(i=1;i<=nd;i++)
      cd[i]=ad[i];
      for(i=1;i<=m;i++)
      a[i]=b[i];
      for(i=1;i<=md;i++)
      ad[i]=bd[i];
      for(i=1;i<=n;i++)
      b[i]=c[i];
      for(i=1;i<=nd;i++)
      bd[i]=cd[i];
      ir=n;
      n=m;
      m=ir;
      ir=nd;
      nd=md;
      md=ir;
m897:;

//c-формирование двух массивов для вычитания
      max=nd;
      if(max<md) max=md;
      if(md==max) goto m78;
      in=md+1;
      for(i=in;i<=max;i++)
      bd[i]=0;
      goto m81;
m78:;
      in=nd+1;
      for(i=in;i<=max;i++)
      ad[i]=0;
m81:;
      it=0;
      in=n+1;
      ik=n+max;
      for(i=in;i<=ik;i++)
      {
      it=it+1;
      a[i]=ad[it];
      }
      it=0;
      in=m+1;
```

```
      ik=m+max;
      for(i=in;i<=ik;i++)
      {
      it=it+1;
      b[i]=bd[it];
      }
      i=0;
      nt=n+max;
      mt=m+max;
m10:;
      if(a[nt]>=b[mt]) goto m1;
      np=nt-1;
m3:;
      if(np==0) goto m201;
      if(a[np]!=0) goto m2;
      np=np-1;
      goto m3;
m2:;
      a[np]=a[np]-1;
m7:;
      np=np+1;
      if(np==nt) goto m30;
      a[np]=9;
      goto m7;
m30:;
      a[nt]=a[nt]+10;
      goto m10;
m1:;
      i=i+1;
      c[i]=a[nt]-b[mt];
      nt=nt-1;
      mt=mt-1;
      if(mt!=0) goto m10;
      if(n==m) goto m781;
m561:;
      i=i+1;
      c[i]=a[nt];
      nt=nt-1;
      if(nt>=1) goto m561;
m781:;
      in=i;
      ik=max+1;
      in=max;
      goto m999;
m201:;
//    write(5,*) 'из меньшего вычитаем большее,здесь нельзя!'
//    exit(0);
      return;
m999:;
      kcra=i-max;
      kdra=max;
      it=0;
      in=i;
      ik=max+1;
      for(j=i;j>=ik;j=j-1)
      {
      it=it+1;
      cra[it]=c[j];
      }
      im=0;
      for(j=1;j<=it;j++)
      {
      if(cra[j]!=0) goto m3122;
```

```
      im=im+1;
      }
      goto m973;
m3122:;
      il=0;
      for(jp=j;jp<=it;jp++)
      {
      il=il+1;
      cra[il]=cra[jp];
      }
m973:;
      kcra=kcra-im;
      it=0;
      in=max;
      for(j=max;j>=1;j=j-1)
      {
      it=it+1;
      dra[it]=c[j];
      }
      if(kcra!=0) goto m971;
      kcra=1;
      cra[1]=0;
m971:;
      if(kdra!=0) goto m972;
      kdra=1;
      dra[1]=0;
m972:;
      if(kdra>krazr) kdra=krazr;
      *pkcra=kcra;
      *pkdra=kdra;
      return;
      }
```

Набор wiczn.c

```
#include<stdio.h>
#include<math.h>
//#include<alloc.h>

      void wiczn(int za,int zb,int *pzc,
       int n,int nd,int *a,int *ad,
      int m,int md,int *b,int *bd,
      int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr)
      {
 extern void sl(int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr);

extern void wich(int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr);

// c-длинное вычитание с учетом знаков уменьшаемого и вычитаемого
 int c[401],cd[401],kcel,kdro,zc,
 ar[401],adr[401],
 ai[401],adi[401],bi[401],bdi[401],zais,zbi,ip,
 zbis,i,ni,ndi,mi,mdi,nmin,nr,ndr;

      zais=za;
      zbis=zb;
      for(i=1;i<=n;i++)
      ai[i]=a[i];
      for(i=1;i<=nd;i++)
      adi[i]=ad[i];
      for(i=1;i<=m;i++)
      bi[i]=b[i];
      for(i=1;i<=md;i++)
      bdi[i]=bd[i];
      ni=n;
      ndi=nd;
      mi=m;
      mdi=md;
      zbi=zb;
      zb=zb*(-1);
      ip=za*zb;
      if(ip>0) goto m781;
      if(za==1) goto m782;
m1901:if(m>n) goto m783;
      if(m==n) goto m784;
      wich(n,nd,a,ad,m,md,b,bd,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=za;
      goto m785;
m784:;
      for(i=1;i<=m;i++)
      {
      if(a[i]>b[i]) goto m787;
      if(a[i]<b[i]) goto m788;
      }
      nmin=nd;
      if(nmin>md) nmin=md;
      for(i=1;i<=nmin;i++)
      {
      if(ad[i]>bd[i]) goto m792;
      if(ad[i]<bd[i]) goto m793;
      }
      if(nd<=md) goto m795;
```

```
      wich(n,nd,a,ad,m,md,b,bd,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=za;
      goto m785;
m795:;
      wich(m,md,b,bd,n,nd,a,ad,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=zb;
      goto m785;
m793:;
      wich(m,md,b,bd,n,nd,a,ad,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=zb;
      goto m785;
m792:;
      wich(n,nd,a,ad,m,md,b,bd,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=za;
      goto m785;
m788:;
      wich(m,md,b,bd,n,nd,a,ad,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=zb;
      goto m785;
m787:;
      wich(n,nd,a,ad,m,md,b,bd,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=za;
      goto m785;
m783:;
      wich(m,md,b,bd,n,nd,a,ad,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=zb;
      goto m785;
m782:;
      for(i=1;i<=n;i++)
      ar[i]=a[i];
      for(i=1;i<=nd;i++)
      adr[i]=ad[i];
      nr=n;
      ndr=nd;
      for(i=1;i<=m;i++)
      a[i]=b[i];
      for(i=1;i<=md;i++)
      ad[i]=bd[i];
      n=m;
      nd=md;
      m=nr;
      md=ndr;
      for(i=1;i<=nr;i++)
      b[i]=ar[i];
      for(i=1;i<=ndr;i++)
      bd[i]=adr[i];
      zc=za;
      za=zb;
      zb=zc;
      goto m1901;
m781:;
      sl(n,nd,a,ad,m,md,b,bd,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=za;
m785:;
      zb=zbi;
      n=ni;
      nd=ndi;
      m=mi;
      md=mdi;
      for(i=1;i<=n;i++)
      a[i]=ai[i];
      for(i=1;i<=nd;i++)
      ad[i]=adi[i];
```

```
for(i=1;i<=m;i++)
b[i]=bi[i];
for(i=1;i<=md;i++)
bd[i]=bdi[i];
za=zais;
zb=zbis;
*pkcel=kcel;
*pkdro=kdro;
return;
}
```

Набор ymnzn.c

```
#include<stdio.h>
#include<math.h>
//#include<alloc.h>
      void ymnzn(int za,int zb,int *pzc,int n,int nd,int *a,int *ad,
      int m,int md,int *b,int *bd,
      int *pkcel,int *pkdro,int *ctot,int *cdtot,int krazr)
      {

 extern void ymn(int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
int *pkcel,int *pkdro,int *ctot,int *cdtot,int krazr);


//c-длинное умножение с учетом знаков сомножителей
      int c[400],cd[400],kcel,kdro,zc,
            csl[400],csld[400],
       rezc[400],rezdr[400],
      ai[400],adi[400],bi[400],bdi[400];
      int zais,zbis,i,ni,ndi,mi,mdi;
      zais=za;
      zbis=zb;
      for(i=1;i<=n;i++)
      ai[i]=a[i];
      for(i=1;i<=nd;i++)

      adi[i]=ad[i];
      for(i=1;i<=m;i++)
      bi[i]=b[i];
      for(i=1;i<=md;i++)
      bdi[i]=bd[i];
      ni=n;
      ndi=nd;
      mi=m;
      mdi=md;

      *pzc=za*zb;

      ymn(n,nd,a,ad,m,md,b,bd,pkcel,pkdro,ctot,cdtot,krazr);

      n=ni;
      nd=ndi;
      m=mi;
      md=mdi;
      for(i=1;i<=n;i++)
      a[i]=ai[i];
      for(i=1;i<=nd;i++)
      ad[i]=adi[i];
      for(i=1;i<=m;i++)
      b[i]=bi[i];
      for(i=1;i<=md;i++)
      bd[i]=bdi[i];
      za=zais;
      zb=zbis;

      return;
      }
```

//c-длинное умножение ymn

```
      void ymn(int n,int nd,int *a,int *ad,
      int m,int md,int *b,int *bd,
      int *pkcel,int *pkdro,int *ctot,int *cdtot,int krazr)
      {

extern void sl(int icel,int idr,int *csl,int *csld,
int ntot,int kdtot,int *ctot,int *cdtot,
int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr);


      int c[400],cd[400],i,it,in,ik,ntot,kdtot,im,p,nt,mt,jp,pr,i1,os,
      kraz,kcd,icel,iab,idr,j,il,kcel,kdro,
      csl[400],csld[400],
      rezc[400],rezdr[400];
      for(i=1;i<=399;i++)
      {
      ctot[i]=0;
      cdtot[i]=0;
      }
//c-формирование двух массивов для умножения
      it=0;
      in=n+1;
      ik=n+nd;
      for(i=in;i<=ik;i++)
      {
      it=it+1;
      a[i]=ad[it];
      }
      it=0;
      in=m+1;
      ik=m+md;
      for(i=in;i<=ik;i++)
      {
      it=it+1;
      b[i]=bd[it];
      }
      ntot=n+m;
      kdtot=nd+md;
      for(i=1;i<=ntot;i++)
      ctot[i]=0;
      for(i=1;i<=kdtot;i++)
      cdtot[i]=0;

//c-умножение
      i=0;
      im=0;
      p=0;
      nt=n+nd;
      mt=m+md;
m801:;
      p=0;
      for(jp=1;jp<=399;jp++)
      c[jp]=0;
      i=0;
      nt=n+nd;
      im=im+1;
m701:;
      pr=a[nt]*b[mt]+p;
      i1=pr/10;
      os=pr-i1*10;
```

```
      i=i+1;
      c[i]=os;
      p=i1;
       nt=nt-1;
      if(nt>=1) goto m701;
      if(p==0) goto m1701;
      i=i+1;
      c[i]=p;
      p=0;
m1701:;
      kraz=md-im+1;
      kcd=nd+kraz;
      icel=i-kcd;
      if(icel<i) goto m1921;
      iab=kcd;
      if(kcd<0)  iab=-kcd;
      it=0;
      for(jp=i;jp>=1;jp=jp-1)
      {
      it=it+1;
      csl[it]=c[jp];
      }
      if(iab==0) goto m1941;
      for(jp=iab;jp>=1;jp=jp-1)
      {
      it=it+1;
      csl[it]=0;
      }
m1941:;
      idr=1;
      csld[1]=0;
      goto m8722;
m1921:;
      it=0;
      in=i;
      ik=kcd+1;
      if(in==ik) goto m961;
      if(in<ik) goto m962;
      for(j=in;j>=ik;j=j-1)
      {
      it=it+1;
      csl[it]=c[j];
      }
      icel=i-kcd;
      goto m963;
m961:;
      icel=1;
      csl[1]=c[in];
      goto m963;
m962:;
      icel=1;
      csl[1]=0;
m963:;
      in=kcd;
      it=0;
      for(j=in;j>=1;j=j-1)
      {
      it=it+1;
      csld[it]=c[j];
      }
      idr=kcd;
      if(idr>0) goto m8722;
      idr=1;
```

```
      csld[1]=0;
m8722:;

      sl(icel,idr,csl,csld,ntot,kdtot,ctot,cdtot,&kcel,&kdro,
     rezc,rezdr,krazr);
      if(kcel!=0) goto m8723;
      kcel=1;
      rezc[1]=0;
m8723:;
      if(kdro!=0) goto m8724;
      kdro=1;
      rezdr[1]=0;
m8724:;
      for(i=1;i<=kcel;i++)
      ctot[i]=rezc[i];
      for(i=1;i<=kdro;i++)
      cdtot[i]=rezdr[i];
      ntot=kcel;
      kdtot=kdro;
      mt=mt-1;
      if(mt>=1) goto m801;
      it=0;
      for(j=1;j<=kcel;j++)
      {
      if(ctot[j]!=0) goto m7767;
      it=it+1;
      }
m7767:;
      il=0;
      for(jp=j;jp<=kcel;jp++)
      {
      il=il+1;
      ctot[il]=ctot[jp];
      }
      kcel=kcel-it;
      if(kcel!=0) goto m3999;
      kcel=1;
      ctot[1]=0;
m3999:;
      if(kdro!=0) goto m4000;
      kdro=1;
      cdtot[1]=0;
m4000:;

      if(kdro>krazr) kdro=krazr;
      *pkcel=kcel;
      *pkdro=kdro;
      return;

      }
```

Набор delzn.c

```
#include<stdio.h>
#include<math.h>
//#include<alloc.h>

//c-длинное деление с учетом знаков
void delzn(int za,int zb,int *pzc,
       int n,int nd,int *a,int *ad,
      int m,int md,int *b,int *bd,
      int *pkcdel,int *pkddel,int *cdel,int *ddel,int krazr)
      {
      extern FILE *fp6;
 extern void deld(int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
int *pkcdel,int *pkddel,int *cdel,int *ddel,int krazr);

      int c[400],cd[400],ni,ndi,mi,mdi,kcdel,kddel,
      ar[400],adr[400],br[400],bdr[400],zais,zbis,i,
      ctot[400],cdtot[400],ld,
      rezc[400],rezdr[400],ce[400],ced[400];
      zais=za;
      zbis=zb;
      for(i=1;i<=n;i++)
      ar[i]=a[i];
      for(i=1;i<=nd;i++)
      adr[i]=ad[i];
      for(i=1;i<=m;i++)
      br[i]=b[i];
      for(i=1;i<=md;i++)
      bdr[i]=bd[i];
      ni=n;
      ndi=nd;
      mi=m;
      mdi=md;
      *pzc=za*zb;
/*      fprintf(fp6,"делимое delzn\n");
      fprintf(fp6,"длина целой части=%d длина дробной части=%d\n", n,nd);
      fprintf(fp6,"\n целая часть\n");
      for(ld=1;ld<=n;ld++)
      fprintf(fp6,"%1d", a[ld]);
      fprintf(fp6,"\n дробная часть\n");
      for(ld=1;ld<=nd;ld++)
      fprintf(fp6,"%1d", ad[ld]);
      fprintf(fp6,"\n делитель  \n");
//    fprintf(fp6,"знак=%d\n",iznt);
      fprintf(fp6,"длина целой части=%d длина дробной части=%d\n",m,md);
      fprintf(fp6,"\n целая часть\n");
      for(ld=1;ld<=m;ld++)
      fprintf(fp6,"%1d", b[ld]);
      fprintf(fp6,"\n дробная часть\n");
      for(ld=1;ld<=md;ld++)
      fprintf(fp6,"%1d", bd[ld]);
      fprintf(fp6,"\n"); */


      deld(n,nd,a,ad,m,md,b,bd,&kcdel,&kddel,cdel,ddel,krazr);
/*    fprintf(fp6,"частное delzn\n");
      fprintf(fp6,"длина целой части=%d длина дробной части=%d\n",kcdel,kddel);
      fprintf(fp6,"\n целая часть\n");
      for(ld=1;ld<=kcdel;ld++)
      fprintf(fp6,"%1d", cdel[ld]);
      fprintf(fp6,"\n дробная часть\n");
```

```
      for(ld=1;ld<=kddel;ld++)
      fprintf(fp6,"%1d",ddel[ld]);
      fprintf(fp6,"\n");        */
      n=ni;
      nd=ndi;
      m=mi;
      md=mdi;
      for(i=1;i<=n;i++)
      a[i]=ar[i];
      for(i=1;i<=nd;i++)
      ad[i]=adr[i];
      for(i=1;i<=m;i++)
      b[i]=br[i];
      for(i=1;i<=md;i++)
      bd[i]=bdr[i];
      za=zais;
      zb=zbis;
      *pkcdel=kcdel;
      *pkddel=kddel;
      return;
      }
```

//deld

```
      void deld(int n,int nd,int *a,int *ad,
       int m,int md,int *b,int *bd,
       int *pkcdel,int *pkddel,int *cdel,int *ddel,int krazr)
       {

extern void wich(int nat,int kad,int *a,int *ad,
int mat,int kbd,int *b,int *bd,
int *pkc,int *pkcd,int *c,int *cd,int krazr);

extern void sl(int kctot,int kdto,int *ctot,int *cdtot,
int kedi,int kedid,int *ce,int *ced,
     int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr);

//c-длинное деление
      int c[400],cd[400],kcdel,kddel,
      ctot[400],cdtot[400],*pkc,*pkcd,*pkcel,*pkdro,
      rezc[400],rezdr[400],ce[400],ced[400];
      int i,max,in,it,ik,il,j,il1,jp,kctot,kdto,
      kedi,kedid,kad,kbd,nat,mat,isd,kcel,kdro,kc,kcd;

      for(i=1;i<=399;i++)
      {

      cdel[i]=0;
       ddel[i]=0;
      }
//c-формирование двух массивов для деления
      max=nd;
      if(max<md) max=md;
      if(md==max) goto m78;
      in=md+1;
      for(i=in;i<=max;i++)
      bd[i]=0;
      goto m81;
m78:;
      in=nd+1;
      for(i=in;i<=max;i++)
      ad[i]=0;
m81:;
      it=0;
```

```
      in=n+1;
      ik=n+max;
      for(i=in;i<=ik;i++)
      {

      it=it+1;
      a[i]=ad[it];
      }

      il=0;
      for(j=1;j<=ik;j++)
      {

      if(a[j]!=0) goto m5672;
      il=il+1;
      }
      kcdel=0;
      kddel=0;
      goto m1231;
//    exit(0);
m5672:;
      it=0;
      for(jp=j;jp<=ik;jp++)
      {

      it=it+1;
      a[it]=a[jp];
      }

      it=0;
      in=m+1;
      ik=m+max;
      for(i=in;i<=ik;i++)
      {

      it=it+1;
      b[i]=bd[it];
      }

      il1=0;
      for(j=1;j<=ik;j++)
      {

      if(b[j]!=0) goto m15672;
      il1=il1+1;
      }
      kcdel=0;
      kddel=0;
      goto m1231;
//    exit(0);
m15672:;
      it=0;
      for(jp=j;jp<=ik;jp++)
      {

      it=it+1;
      b[it]=b[jp];
      }


//c        if(n.lt.m) goto 675
//c   if(n.eq.m) goto 676
      for(i=1;i<=399;i++)
```

```
      {
      ctot[i]=0;
      cdtot[i]=0;
      }
      kctot=1;
      kdto=1;
      ce[1]=1;
      kedi=1;
      ced[1]=0;
      kedid=1;
      kad=1;
      kbd=1;
      ad[1]=0;
      bd[1]=0;
      nat=n+max-il;
      mat=m+max-il1;
      isd=0;
      kcel=1;
      kdro=1;
      rezc[1]=0;
      rezdr[1]=0;
m2010:;
      if(nat<mat) goto m675;
      if(nat==mat) goto m676;
      goto m677;
m675:;
      if(isd!=0) goto m5621;
      for(j=1;j<=kcel;j++)
      cdel[j]=rezc[j];
      for(j=1;j<=kdro;j++)
      ddel[j]=rezdr[j];
      rezc[1]=0;
      kcdel=kcel;
      kddel=kdro;
      if(nat==0) goto m1231;
m5621:;
      if(isd==0) goto m8721;
      ddel[isd]=rezc[1];
      rezc[1]=0;
      kddel=isd;
m8721:;
      if(nat==0) goto m1231;
      for(j=1;j<=nat;j++)
      {
      if(a[j]!=0) goto m9782;
      }
      goto m1231;
m9782:;
      nat=nat+1;
      isd=isd+1;

      if(isd>krazr) goto m1231;
      a[nat]=0;
      for(i=1;i<=399;i++)
      {
      ctot[i]=0;
      cdtot[i]=0;
      }
      kctot=1;
      kdto=1;
      goto m2010;
m676:;
      for(j=1;j<=nat;j++)
```

```
       {
       if(a[j]>b[j]) goto m677;
       if(a[j]<b[j]) goto m675;
       }
m677:;
       wich(nat,kad,a,ad,mat,kbd,b,bd,&kc,&kcd,c,cd,krazr);
       sl(kctot,kdto,ctot,cdtot,kedi,kedid,ce,ced,
      &kcel,&kdro,rezc,rezdr,krazr);
       kctot=kcel;
       kdto=kdro;
       for(i=1;i<=kcel;i++)
       ctot[i]=rezc[i];
       for(i=1;i<=kdro;i++)
       cdtot[i]=rezdr[i];
       nat=kc;
       kad=kcd;
       if(nat==0) a[1]=0;
       for(j=1;j<=nat;j++)
       a[j]=c[j];
       for(j=1;j<=kad;j++)
       ad[j]=cd[j];
       goto m2010;
m1231:;
       if(kcdel!=0) goto m971;
       kcdel=1;
       cdel[1]=0;
m971:;
       if(kddel!=0) goto m972;
       kddel=1;
       ddel[1]=0;
m972:;
       if(kddel>krazr) kddel=krazr;
       *pkcdel=kcdel;
       *pkddel=kddel;
       return;
       }
```

ПРИЛОЖЕНИЕ-3(набор pidlmpi.c для реализации распараллеливания процедуры суммирования)

Набор pidlmpi.c

```
#include"mpi.h"
#include<math.h>
#include<stdio.h>

int main( int argc, char *argv[ ] )
       {
extern void delzn(int izr,int iznt,int *pizrez,int ncr,
    int ndr,int *rc,int *rd,
    int nct,int ndt,int *tc,int *td,
     int *pkcrez,int *pkdrez,int *rezc,int *rezd,int krazr);

     extern void ymnzn(int izp,int izr,int *pizrez,int ncp,int ndp,
int *pc,int *pd,
int ncr,int ndr,int *rc,int *rd,
     int *pkcrez,int *pkdrez,int *rezc,int *rezd,int krazr);

extern void wiczn(int izr1,int izr,int *pizrez,int ncr1,int ndr1,
int *rc1,int *rd1,
int ncr,int ndr,int *rc,int *rd,
int *pkcrez,int *pkdrez,int *rezc,int *rezd,int krazr);

 extern void slzn(int izsum,int izn2,int *pizrez,int isc,int isd,
int *icsum,int *idsum,
int ikc2,int ikd2,
     int *imc2,int *imd2,
int *pkcrez,int *pkdrez,int *icrez,int *idrez,int krazr);


 int myid, numprocs,izkop,kckop,kdkop,peres[401],nper;
    MPI_Status status;

 MPI_Init(&argc, &argv);
 MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
 MPI_Comm_rank(MPI_COMM_WORLD,&myid);

      int ixc[401],ixd[401],isuc[401],isud[401],idxc[401],ios,l,
      idxd[401],icrez[401],idrez[401],icr1[401],ikp,irab[401],
      idr1[401],icr2[401],idr2[401],iced[401],ided[401],
      ic4[401],id4[401],icpr[401],idpr[401],ickop[401],
      idkop[401],idxn[401],icxn[401];
      int izrez,ic1,id1,iznak,ic2,id2,kcrez,kdrez,iz1,
      izdel,izymn,izsum;
      int izn1,kced,kded,izndx,kcd,i,izn4,kc4,kd4,iznx,kcx,
      kdx,iznax,kcxn,kdxn,iznsu,kcsu,kdsu,isch,iznpr,kcpr,
      kdpr,kdd,krazr,kproc;
      int izr,iznt,*pizrez, ncr,iztot,kctot,kdtot,ictot[401],idtot[401],
      ipa,ipos,tag,
    ndr,rc[401], rd[401],
    nct,ndt,tc[401],td[401],
     *pkcrez,*pkdrez, rezc[401], rezd[401];
      int izp,    ncp,  ndp,
  pc[401],  pd[401];
int izr1,   ncr1,   ndr1,
   rc1[401],   rd1[401];
int    izn2,    isc,   isd,
   icsum[401],   idsum[401],
```

```
   ikc2,   ikd2,
        imc2[401],   imd2[401];
//    icrez[401],   idrez[401];


//    open(6,file='pipardl',status='Unknown')
/* c-kcx-количество разрядов в целой части x
c-kdx-количество разрядов в дробной части x
c-kcsu-количество разрядов в целой части интегральной суммы
c-kdsu-количество разрядов в дробной части интегральной суммы
c-ixc-массив разрядов целой части x
c-ixd-массив разрядов дробной части x
c-isuc-массив разрядов целой части интегральной суммы
c-isud-массив разрядов дробной части интегральной суммы
c-idxc-массив разрядов целой части dx
c-idxd-массив разрядов дробной части dx
c-kcd-количество разрядов в целой части dx
c-kdd-количество разрядов в дробной части dx
c-iced-массив разрядов целой части 1(единицы)
c-ided-массив разрядов дробной части 1(единицы)
c-kced-количество разрядов в целой части 1(единицы)
c-kded-количество разрядов в дробной части 1(единицы) */
      kdd=5;krazr=20;kproc=numprocs-1;
      if(myid==0)
      {
      printf("количество разрядов в шаге dx=%d\n",kdd);
   printf("количество разрядов в дробной части=%d\n",krazr);
   printf("количество процессов=%d\n",kproc);


      MPI_Bcast(&kdd, 1, MPI_INT, 0, MPI_COMM_WORLD);
      MPI_Bcast(&krazr, 1, MPI_INT, 0, MPI_COMM_WORLD);

      MPI_Bcast(&numprocs, 1, MPI_INT, 0, MPI_COMM_WORLD);
      MPI_Bcast(&kproc, 1, MPI_INT, 0, MPI_COMM_WORLD);


// priem
//c- определяем dx
       izndx=1;
       kcd=1;
//c   kdd=2
      idxc[1]=0;
//    do 1 i=1,kdd
      for(i=1;i<=kdd;i++)

      idxd[i]=0;
      idxd[kdd]=1;

//c-определяем 4(четыре)
       izn4=1;
       kc4=1;
       kd4=1;
      ic4[1]=4;
      id4[1]=0;
// определение интегральной суммы от процессов
      iztot=1;
      kctot=1;
      kdtot=1;
      ictot[1]=0;
      idtot[1]=0;
      for(ipa=1;ipa<numprocs;ipa++)
      {
```

```
      ipos=0;

      ipos=ipos+1;
      MPI_Recv(&iznsu,1,MPI_INT,ipa,ipos, MPI_COMM_WORLD,&status);
      ipos=ipos+1;
      MPI_Recv(&kcsu,1,MPI_INT,ipa,ipos, MPI_COMM_WORLD,&status);
      ipos=ipos+1;
      MPI_Recv(&kdsu,1,MPI_INT,ipa,ipos, MPI_COMM_WORLD,&status);

      ipos=ipos+1;
      nper=kcsu+1;


       MPI_Recv(peres,nper,MPI_INT,ipa,ipos, MPI_COMM_WORLD,&status);
       for(i=1;i<=kcsu;i++)
      isuc[i]= peres[i];


      ipos=ipos+1;
      nper=kdsu+1;


       MPI_Recv(peres,nper,MPI_INT,ipa,ipos, MPI_COMM_WORLD,&status);
      for(i=1;i<=kdsu;i++)
      isud[i]= peres[i];
// накопление интегральной суммы от процессов
      slzn(iztot,iznsu,&izrez,
      kctot,kdtot,ictot,idtot,
      kcsu,kdsu,isuc,isud,
      &ic1,&id1,icr1,idr1,krazr);
// переопределение интегральной суммы от процессов
      iztot=izrez;
      kctot=ic1;
      kdtot=id1;
      for(i=1;i<=kctot;i++)
      ictot[i]=icr1[i];
      for(i=1;i<=kdtot;i++)
      idtot[i]=idr1[i];
      }/* end po ipa */
      m11:;
//c-уьножение интегральной суммы на 4
      ymnzn(iztot,izn4,&izrez,
     kctot,kdtot,ictot,idtot,
     kc4,kd4,ic4,id4,
     &kcrez,&kdrez,icrez,idrez,krazr);
//c-вывод числа пи
//c   open(6,file='ch_pi',status='Unknown')
//    write(6,41)
      printf("шаг интегрирования\n");
//    write(6,42)
      printf("целая часть шага\n");
//    write(6,34)
      for(i=1;i<=kcd;i++)
      printf("%1d",idxc[i]);
      printf("\n");
//    write(6,43)
      printf("дробная часть шага\n");
//    write(6,34)
      for(i=1;i<=kdd;i++)
      printf("%1d",idxd[i]);
      printf("\n");
//    write(6,31)
      printf("число пи\n");
```

```
//    write(6,33)
//    write(*,33)
      printf("целая часть\n");
//    write(6,34) (icrez(i),i=1,kcrez)
      for(i=1;i<=kcrez;i++)
      printf("%1d",icrez[i]);
      printf("\n");
//34  format(/1x,70i1)
       printf("дробная часть\n");
//    write(6,34) (idrez(i),i=1,kdrez)
      for(i=1;i<=kdrez;i++)
      printf("%1d",idrez[i]);
      printf("\n");

      }/* myid==0*/


       else
      { /* myid!=0 */

//    for(ipa=1;ipa<numprocs;ipa++)
//    {
      ipa=myid;/* ipa-номер процесса */
//c-количество процессов
      izkop=1;
      kdkop=1;

//c-формирование массива целой части количества процессов
      ikp=0;
m2001:;
      ios=kproc-kproc/10*10;
      ikp=ikp+1;
      irab[ikp]=ios;
      kproc=kproc/10;
      if(kproc!=0) goto m2001;
      kckop=ikp;
      l=0;
      for(i=ikp;i>=1;i=i-1)
      {
      l=l+1;
      ickop[l]=irab[i];
      }

      idkop[1]=0;


      tag=1;
      ipos=0;
//c-определяем 1(единицу)
       izn1=1;
       kced=1;
       kded=1;
      iced[1]=1;
      ided[1]=0;
//c- определяем dx
       izndx=1;
       kcd=1;
//c   kdd=2
      idxc[1]=0;
//    do 1 i=1,kdd
      for(i=1;i<=kdd;i++)

      idxd[i]=0;
```

```
      idxd[kdd]=1;
//c-определяем 4(четыре)
       izn4=1;
       kc4=1;
       kd4=1;
      ic4[1]=4;
      id4[1]=0;
//c-определяем начальное x(x=0)
       iznx=1;
       kcx=1;
       kdx=1;
      ixc[1]=0;
      ixd[1]=0;
//c-меняющееся начальное x
       iznax=1;
       kcxn=1;
       kdxn=1;
      icxn[1]=0;
      idxn[1]=0;
//c-определяем начальное значение интегральной суммы
       iznsu=1;
       kcsu=1;
       kdsu=1;
      isuc[1]=0;
      isud[1]=0;
       isch=0;
//m22:;
//c-определяем номер процесса
       iznpr=1;/* знак номера процесса */
       kdpr=1;/* количество разрядов в дробной части номера процесса */
//c-формирование массива целой части номера процесса
      ikp=0;
n2001:;
      ios=ipa-ipa/10*10;
      ikp=ikp+1;
      irab[ikp]=ios;
      ipa=ipa/10;
      if(ipa!=0) goto n2001;
      kcpr=ikp;/* количество разрядов в целой части номера процесса */
      l=0;
      for(i=ikp;i>=1;i=i-1)
      {
      l=l+1;
      icpr[l]=irab[i];
      }
      idpr[1]=0;
m777:;
//c-вычитаем из номера процесса единицу
       wiczn(iznpr,izn1,&izrez,
     kcpr,kdpr,icpr,idpr,
     kced,kded,iced,ided,
     &ic1,&id1,icr1,idr1,krazr);
//c-умножаем (npr-1)*dx
      ymnzn(izrez,izndx,&iznak,
     ic1,id1,icr1,idr1,
     kcd,kdd,idxc,idxd,
     &ic2,&id2,icr2,idr2,krazr);
//c-складываем x+(npr-1)*dx
      slzn(iznax,iznak,&izrez,
     kcxn,kdxn,icxn,idxn,
     ic2,id2,icr2,idr2,
     &ic1,&id1,icr1,idr1,krazr);
//c-переопределение x
```

```
      iznx=izrez;
      kcx=ic1;
      kdx=id1;
//    do 210 i=1,kcx
      for(i=1;i<=kcx;i++)

      ixc[i]=icr1[i];
      for(i=1;i<=kdx;i++)
      ixd[i]=idr1[i];
      isch=isch+1;
      if(isch<1000) goto m678;
      if(ixc[1]>=iced[1]) goto m1111;
//c   write(6,891) icpr(1)
//891 format(/1x,'номер процесса=',i3/)
//    write(*,*) 'дробная часть x'
//    write(*,34) (ixd(l),l=1,kdx)
      isch=0;
m678:;
//c-сравниваем x с единицей
      if(ixc[1]>=iced[1]) goto m1111;

//c-умножаем x на x
      ymnzn(iznx,iznx,&izrez,

     kcx,kdx,ixc,ixd,
     kcx,kdx,ixc,ixd,
     &kcrez,&kdrez,icrez,idrez,krazr);
//c-складываем 1+x*x
      slzn(izn1,izrez,&iz1,
     kced,kded,iced,ided,
     kcrez,kdrez,icrez,idrez,
     &ic1,&id1,icr1,idr1,krazr);
//c-делим 1/(1+x*x)
      delzn(izn1,iz1,&izdel,
     kced,kded,iced,ided,
     ic1,id1,icr1,idr1,
     &ic2,&id2,icr2,idr2,krazr);
//c-умножаем 1/(1+x*x) на dx
      ymnzn(izdel,izndx,&izymn,
     ic2,id2,icr2,idr2,
     kcd,kdd,idxc,idxd,
     &ic1,&id1,icr1,idr1,krazr);
//c-наращиваем интегральную сумму
      slzn(iznsu,izymn,&izsum,
     kcsu,kdsu,isuc,isud,
     ic1,id1,icr1,idr1,
     &ic2,&id2,icr2,idr2,krazr);
//c-переопределяем интегральную сумму
      iznsu=izsum;
      kcsu=ic2;
      kdsu=id2;
      for(i=1;i<=ic2;i++)
      isuc[i]=icr2[i];
      for(i=1;i<=id2;i++)
      isud[i]=idr2[i];
//c-увеличить меняющееся начальное x на kproc*dx
//c-вычислить kproc*dx
      ymnzn(izkop,izndx,&izrez,
     kckop,kdkop,ickop,idkop,
     kcd,kdd,idxc,idxd,
     &ic1,&id1,icr1,idr1,krazr);
//c-суммируем меняющееся начальное x+kproc*dx
      slzn(iznax,izrez,&iz1,
```

```
     kcxn,kdxn,icxn,idxn,
     ic1,id1,icr1,idr1,
     &ic2,&id2,icr2,idr2,krazr);
//c-переопределяем меняющееся начальное x
      iznax=iz1;
      kcxn=ic2;
      kdxn=id2;
      for(i=1;i<=kcxn;i++)
      icxn[i]=icr2[i];
      for( i=1;i<=kdxn;i++)
      idxn[i]=idr2[i];
      goto m777;
m1111:;
//-посылку в нулевой процесс интегральной суммы сделать
      peres[0]=0;
      ipos=ipos+1;
      MPI_Send(&iznsu,1,MPI_INT,0,ipos, MPI_COMM_WORLD);
      ipos=ipos+1;
      MPI_Send(&kcsu,1,MPI_INT,0,ipos, MPI_COMM_WORLD);
      ipos=ipos+1;
      MPI_Send(&kdsu,1,MPI_INT,0,ipos, MPI_COMM_WORLD);
      for(i=1;i<=kcsu;i++)
      peres[i]=isuc[i];
      ipos=ipos+1;
      nper=kcsu+1;


       MPI_Send(peres,nper,MPI_INT,0,ipos, MPI_COMM_WORLD);
      for(i=1;i<=kdsu;i++)
      peres[i]=isud[i];
      ipos=ipos+1;
      nper=kdsu+1;


      MPI_Send(peres,nper,MPI_INT,0,ipos, MPI_COMM_WORLD);

      }/* myid!=0*/

      MPI_Finalize();
      }

//#include<stdio.h>
//#include<math.h>
//#include<alloc.h>

//c-длинное деление с учетом знаков
void delzn(int za,int zb,int *pzc,
       int n,int nd,int *a,int *ad,
      int m,int md,int *b,int *bd,
      int *pkcdel,int *pkddel,int *cdel,int *ddel,int krazr)
      {
//    extern FILE *fp6;
 extern void deld(int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
int *pkcdel,int *pkddel,int *cdel,int *ddel,int krazr);

      int c[400],cd[400],ni,ndi,mi,mdi,kcdel,kddel,
      ar[400],adr[400],br[400],bdr[400],zais,zbis,i,
      ctot[400],cdtot[400],ld,
      rezc[400],rezdr[400],ce[400],ced[400];
      zais=za;
      zbis=zb;
      for(i=1;i<=n;i++)
```

```
      ar[i]=a[i];
      for(i=1;i<=nd;i++)
      adr[i]=ad[i];
      for(i=1;i<=m;i++)
      br[i]=b[i];
      for(i=1;i<=md;i++)
      bdr[i]=bd[i];
      ni=n;
      ndi=nd;
      mi=m;
      mdi=md;
      *pzc=za*zb;
/*      fprintf(fp6,"делимое delzn\n");
      fprintf(fp6,"длина целой части=%d длина дробной части=%d\n", n,nd);
      fprintf(fp6,"\n целая часть\n");
      for(ld=1;ld<=n;ld++)
      fprintf(fp6,"%1d", a[ld]);
      fprintf(fp6,"\n дробная часть\n");
      for(ld=1;ld<=nd;ld++)
      fprintf(fp6,"%1d", ad[ld]);
      fprintf(fp6,"\n делитель  \n");
//    fprintf(fp6,"знак=%d\n",iznt);
      fprintf(fp6,"длина целой части=%d длина дробной части=%d\n",m,md);
      fprintf(fp6,"\n целая часть\n");
      for(ld=1;ld<=m;ld++)
      fprintf(fp6,"%1d", b[ld]);
      fprintf(fp6,"\n дробная часть\n");
      for(ld=1;ld<=md;ld++)
      fprintf(fp6,"%1d", bd[ld]);
      fprintf(fp6,"\n"); */


      deld(n,nd,a,ad,m,md,b,bd,&kcdel,&kddel,cdel,ddel,krazr);
/*    fprintf(fp6,"частное delzn\n");
      fprintf(fp6,"длина целой части=%d длина дробной части=%d\n",kcdel,kddel);
      fprintf(fp6,"\n целая часть\n");
      for(ld=1;ld<=kcdel;ld++)
      fprintf(fp6,"%1d", cdel[ld]);
      fprintf(fp6,"\n дробная часть\n");
      for(ld=1;ld<=kddel;ld++)
      fprintf(fp6,"%1d",ddel[ld]);

      fprintf(fp6,"\n");        */
      n=ni;
      nd=ndi;
      m=mi;
      md=mdi;
      for(i=1;i<=n;i++)
      a[i]=ar[i];
      for(i=1;i<=nd;i++)
      ad[i]=adr[i];
      for(i=1;i<=m;i++)
      b[i]=br[i];
      for(i=1;i<=md;i++)
      bd[i]=bdr[i];
      za=zais;
      zb=zbis;
      *pkcdel=kcdel;
      *pkddel=kddel;
      return;
      }
//deld
      void deld(int n,int nd,int *a,int *ad,
```

```
        int m,int md,int *b,int *bd,
        int *pkcdel,int *pkddel,int *cdel,int *ddel,int krazr)
        {

extern void wich(int nat,int kad,int *a,int *ad,
int mat,int kbd,int *b,int *bd,
int *pkc,int *pkcd,int *c,int *cd,int krazr);

extern void sl(int kctot,int kdto,int *ctot,int *cdtot,
int kedi,int kedid,int *ce,int *ced,
     int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr);

//c-длинное деление
      int c[400],cd[400],kcdel,kddel,
      ctot[400],cdtot[400],*pkc,*pkcd,*pkcel,*pkdro,
      rezc[400],rezdr[400],ce[400],ced[400];
      int i,max,in,it,ik,il,j,il1,jp,kctot,kdto,
      kedi,kedid,kad,kbd,nat,mat,isd,kcel,kdro,kc,kcd;

      for(i=1;i<=399;i++)
      {

      cdel[i]=0;
       ddel[i]=0;
      }
//c-формирование двух массивов для деления
      max=nd;
      if(max<md) max=md;
      if(md==max) goto m78;
      in=md+1;
      for(i=in;i<=max;i++)
      bd[i]=0;
      goto m81;
m78:;
      in=nd+1;
      for(i=in;i<=max;i++)
      ad[i]=0;
m81:;
      it=0;
      in=n+1;
      ik=n+max;
      for(i=in;i<=ik;i++)
      {

      it=it+1;
      a[i]=ad[it];
      }

      il=0;
      for(j=1;j<=ik;j++)
      {

      if(a[j]!=0) goto m5672;
      il=il+1;
      }
      kcdel=0;
      kddel=0;
      goto m1231;
//    exit(0);
m5672:;
      it=0;
      for(jp=j;jp<=ik;jp++)
      {
```

```
      it=it+1;
      a[it]=a[jp];
      }

      it=0;
      in=m+1;
      ik=m+max;
      for(i=in;i<=ik;i++)
      {

      it=it+1;
      b[i]=bd[it];
      }

      il1=0;
      for(j=1;j<=ik;j++)
      {

      if(b[j]!=0) goto m15672;
      il1=il1+1;
      }
      kcdel=0;
      kddel=0;
      goto m1231;
//    exit(0);
m15672:;
      it=0;
      for(jp=j;jp<=ik;jp++)
      {

      it=it+1;
      b[it]=b[jp];
      }


//c        if(n.lt.m) goto 675
//c   if(n.eq.m) goto 676
      for(i=1;i<=399;i++)
      {
      ctot[i]=0;
      cdtot[i]=0;
      }
      kctot=1;
      kdto=1;
      ce[1]=1;

      kedi=1;
      ced[1]=0;
      kedid=1;
      kad=1;
      kbd=1;
      ad[1]=0;
      bd[1]=0;
      nat=n+max-il;
      mat=m+max-il1;
      isd=0;
      kcel=1;
      kdro=1;
      rezc[1]=0;
      rezdr[1]=0;
m2010:;
      if(nat<mat) goto m675;
```

```
      if(nat==mat) goto m676;
      goto m677;
m675:;
      if(isd!=0) goto m5621;
      for(j=1;j<=kcel;j++)
      cdel[j]=rezc[j];
      for(j=1;j<=kdro;j++)
      ddel[j]=rezdr[j];
      rezc[1]=0;
      kcdel=kcel;
      kddel=kdro;
      if(nat==0) goto m1231;
m5621:;
      if(isd==0) goto m8721;
      ddel[isd]=rezc[1];
      rezc[1]=0;
      kddel=isd;
m8721:;
      if(nat==0) goto m1231;
      for(j=1;j<=nat;j++)
      {
      if(a[j]!=0) goto m9782;
      }
      goto m1231;
m9782:;
      nat=nat+1;
      isd=isd+1;

      if(isd>krazr) goto m1231;
      a[nat]=0;
      for(i=1;i<=399;i++)
      {
      ctot[i]=0;
      cdtot[i]=0;
      }
      kctot=1;
      kdto=1;
      goto m2010;
m676:;
      for(j=1;j<=nat;j++)
      {
      if(a[j]>b[j]) goto m677;
      if(a[j]<b[j]) goto m675;

      }
m677:;
      wich(nat,kad,a,ad,mat,kbd,b,bd,&kc,&kcd,c,cd,krazr);
      sl(kctot,kdto,ctot,cdtot,kedi,kedid,ce,ced,
     &kcel,&kdro,rezc,rezdr,krazr);
      kctot=kcel;
      kdto=kdro;
      for(i=1;i<=kcel;i++)
      ctot[i]=rezc[i];
      for(i=1;i<=kdro;i++)
      cdtot[i]=rezdr[i];

      nat=kc;
      kad=kcd;
      if(nat==0) a[1]=0;
      for(j=1;j<=nat;j++)
      a[j]=c[j];
      for(j=1;j<=kad;j++)
      ad[j]=cd[j];
```

```
      goto m2010;
m1231:;
      if(kcdel!=0) goto m971;
      kcdel=1;
      cdel[1]=0;
m971:;
      if(kddel!=0) goto m972;
      kddel=1;
      ddel[1]=0;
m972:;

      if(kddel>krazr) kddel=krazr;
      *pkcdel=kcdel;
      *pkddel=kddel;
      return;
      }

//#include<stdio.h>
//#include<math.h>
//#include<alloc.h>

//c-длинное сложение
      void sl(int n,int nd,int *a,int *ad,
  int m,int md,int *b,int *bd,
int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr)
{
int c[401],cd[401],i,j,ir,max,in,kcel,kdro,
it,ik,p,nt,mt,rez,i1,os;

        for(i=1;i<=399;i++)
      {
      rezc[i]=0;
      rezdr[i]=0;
      }
      if(n>=m) goto m897;
      for(i=1;i<=n;i++)
      c[i]=a[i];
      for(i=1;i<=nd;i++)
      cd[i]=ad[i];
      for(i=1;i<=m;i++)
      a[i]=b[i];
      for(i=1;i<=md;i++)
      ad[i]=bd[i];
      for(i=1;i<=n;i++)
      b[i]=c[i];
      for(i=1;i<=nd;i++)
      bd[i]=cd[i];
      ir=n;
      n=m;
      m=ir;
      ir=nd;
      nd=md;
      md=ir;
m897:;
//c-формирование двух массивов для сложения
      max=nd;
      if(max<md) max=md;
      if(md==max) goto m78;
      in=md+1;
      for(i=in;i<=max;i++)
      bd[i]=0;
      goto m81;
m78:;
```

```
      in=nd+1;
      for(i=in;i<=max;i++)
      ad[i]=0;
m81:;
      it=0;
      in=n+1;
      ik=n+max;
      for(i=in;i<=ik;i++)
      {
      it=it+1;
      a[i]=ad[it];
      }
      it=0;
      in=m+1;
      ik=m+max;
      for(i=in;i<=ik;i++)
      {
      it=it+1;
      b[i]=bd[it];
      }

//c-сложение
      i=0;
      p=0;

      nt=n+max;
      mt=m+max;
m7:;
      if(mt<1) rez=a[nt];
      if(mt>=1) rez=a[nt]+b[mt];
      rez=rez+p;
      i1=rez/10;
      os=rez-i1*10;
      i=i+1;
      c[i]=os;
//c   if(i.ne.1) goto 1
      p=i1;
      nt=nt-1;
      mt=mt-1;
      if((p==0)&&(mt<1)) goto m4;
      if((nt<1)&&(p!=0)) goto m5;
      if((nt<1)&&(p==0)) goto m4;
      goto m7;
m5:;
      i=i+1;
      c[i]=p;
      goto m67;
m4:;

m89:;
      if(nt<1) goto m67;
      i=i+1;
      c[i]=a[nt];
      nt=nt-1;
      goto m89;
m67:;
      ik=max+1;
      kcel=i-max;
      kdro=max;
      it=0;
      for(j=i;j>=ik;j=j-1)
      {
      it=it+1;
```

```
      rezc[it]=c[j];
      }
      it=0;
      for(j=max;j>=1;j=j-1)
      {
      it=it+1;
      rezdr[it]=c[j];
      }
      if(kcel!=0) goto m971;
      kcel=1;
      rezc[1]=0;
m971:;
      if(kdro!=0) goto m972;
      kdro=1;
      rezdr[1]=0;
m972:;
      if(kdro>krazr) kdro=krazr;
      *pkcel=kcel;
      *pkdro=kdro;
      return;
      }


//#include<stdio.h>
//#include<math.h>
//#include<alloc.h>

      void slzn(int za,int zb,int *pzc,
 int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
     int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr)
{

 extern void sl(int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,

int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr);

extern void wich(int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr);

//c-длинное сложение с учетом знаков слагаемых

int c[401],cd[401],kcel,kdro,
ar[401],adr[401],zc,
ai[401],adi[401],bi[401],bdi[401];
int zais,zbis,i,ni,ndi,mi,mdi,ip,nmin,nr,ndr;
      zais=za;
      zbis=zb;
      for(i=1;i<=n;i++)
      ai[i]=a[i];

      for(i=1;i<=nd;i++)
      adi[i]=ad[i];
      for(i=1;i<=m;i++)
      bi[i]=b[i];
      for(i=1;i<=md;i++)
      bdi[i]=bd[i];
      ni=n;
      ndi=nd;
      mi=m;
      mdi=md;
```

```
      ip=za*zb;
      if(ip>0) goto m781;
      if(za==1) goto m782;
m1901:      if(m>n) goto m783;
      if(m==n) goto m784;
      wich(n,nd,a,ad,m,md,b,bd,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=za;
      goto m785;
m784:;
      for(i=1;i<=m;i++)
      {
      if(a[i]>b[i]) goto m787;
      if(a[i]<b[i]) goto m788;
      }
      nmin=nd;
      if(nmin>md) nmin=md;
      for(i=1;i<=nmin;i++)
      {
      if(ad[i]>bd[i]) goto m792;
      if(ad[i]<bd[i]) goto m793;
      }
      if(nd<=md) goto m795;
      wich(n,nd,a,ad,m,md,b,bd,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=za;
      goto m785;
m795:;
      wich(m,md,b,bd,n,nd,a,ad,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=zb;
      goto m785;
m793:;
      wich(m,md,b,bd,n,nd,a,ad,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=zb;
      goto m785;
m792:;

      wich(n,nd,a,ad,m,md,b,bd,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=za;
      goto m785;
m788:;
      wich(m,md,b,bd,n,nd,a,ad,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=zb;
      goto m785;
m787:;
      wich(n,nd,a,ad,m,md,b,bd,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=za;
      goto m785;
m783:;
      wich(m,md,b,bd,n,nd,a,ad,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=zb;
      goto m785;
m782:;
      for(i=1;i<=n;i++)
      ar[i]=a[i];

      for(i=1;i<=nd;i++)
      adr[i]=ad[i];
      nr=n;
      ndr=nd;
      for(i=1;i<=m;i++)
      a[i]=b[i];
      for(i=1;i<=md;i++)
      ad[i]=bd[i];
```

```
      n=m;
      nd=md;
      m=nr;
      md=ndr;
      for(i=1;i<=nr;i++)
      b[i]=ar[i];
      for(i=1;i<=ndr;i++)
      bd[i]=adr[i];
      zc=za;
      za=zb;
      zb=zc;
      goto m1901;
m781:;
      sl(n,nd,a,ad,m,md,b,bd,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=za;
m785:;
      n=ni;
      nd=ndi;
      m=mi;
      md=mdi;
      for(i=1;i<=n;i++)
      a[i]=ai[i];
      for(i=1;i<=nd;i++)
      ad[i]=adi[i];
      for(i=1;i<=m;i++)
      b[i]=bi[i];
      for(i=1;i<=md;i++)
      bd[i]=bdi[i];

      za=zais;
      zb=zbis;
      *pkcel=kcel;
      *pkdro=kdro;
      return;
      }


//#include<stdio.h>
//#include<math.h>
//#include<alloc.h>

//c-длинное вычитание
      void wich(int n,int nd,int *a,int *ad,
 int m,int md,int *b,int *bd,
 int *pkcra,int *pkdra,int *cra,int *dra,int krazr)
      {
int c[401],cd[401],i,ir,max,jp,kcra,kdra,
in,it,ik,nt,mt,np,il,j,im;

      for(i=1;i<=399;i++)
      {
      cra[i]=0;
      dra[i]=0;
      }
      if(n>=m) goto m897;
      for(i=1;i<=n;i++)
      c[i]=a[i];
      for(i=1;i<=nd;i++)
      cd[i]=ad[i];
      for(i=1;i<=m;i++)
      a[i]=b[i];
      for(i=1;i<=md;i++)
      ad[i]=bd[i];
```

```
      for(i=1;i<=n;i++)
      b[i]=c[i];
      for(i=1;i<=nd;i++)
      bd[i]=cd[i];
      ir=n;
      n=m;
      m=ir;
      ir=nd;
      nd=md;
      md=ir;
m897:;

//c-формирование двух массивов для вычитания
      max=nd;
      if(max<md) max=md;
      if(md==max) goto m78;
      in=md+1;
      for(i=in;i<=max;i++)
      bd[i]=0;
      goto m81;
m78:;
      in=nd+1;
      for(i=in;i<=max;i++)
      ad[i]=0;
m81:;
      it=0;
      in=n+1;
      ik=n+max;
      for(i=in;i<=ik;i++)
      {
      it=it+1;
      a[i]=ad[it];
      }
      it=0;
      in=m+1;
      ik=m+max;
      for(i=in;i<=ik;i++)
      {
      it=it+1;
      b[i]=bd[it];
      }
      i=0;
      nt=n+max;
      mt=m+max;
m10:;
      if(a[nt]>=b[mt]) goto m1;
      np=nt-1;
m3:;
      if(np==0) goto m201;
      if(a[np]!=0) goto m2;
      np=np-1;
      goto m3;
m2:;
      a[np]=a[np]-1;
m7:;
      np=np+1;
      if(np==nt) goto m30;
      a[np]=9;
      goto m7;
m30:;
      a[nt]=a[nt]+10;
      goto m10;
m1:;
```

```
      i=i+1;
      c[i]=a[nt]-b[mt];
      nt=nt-1;
      mt=mt-1;
      if(mt!=0) goto m10;
      if(n==m) goto m781;
m561:;
      i=i+1;
      c[i]=a[nt];
      nt=nt-1;
      if(nt>=1) goto m561;
m781:;
      in=i;
      ik=max+1;
      in=max;
      goto m999;
m201:;
//    write(5,*) 'из меньшего вычитаем большее,здесь нельзя!'
//    exit(0);

      return;
m999:;
      kcra=i-max;
      kdra=max;
      it=0;
      in=i;
      ik=max+1;
      for(j=i;j>=ik;j=j-1)
      {
      it=it+1;
      cra[it]=c[j];
      }
      im=0;
      for(j=1;j<=it;j++)
      {
      if(cra[j]!=0) goto m3122;
      im=im+1;
      }
      goto m973;
m3122:;
      il=0;
      for(jp=j;jp<=it;jp++)
      {
      il=il+1;
      cra[il]=cra[jp];
      }
m973:;
      kcra=kcra-im;
      it=0;
      in=max;
      for(j=max;j>=1;j=j-1)
      {
      it=it+1;
      dra[it]=c[j];
      }
      if(kcra!=0) goto m971;
      kcra=1;
      cra[1]=0;
m971:;
      if(kdra!=0) goto m972;
      kdra=1;
      dra[1]=0;
m972:;
```

```
      if(kdra>krazr) kdra=krazr;
      *pkcra=kcra;
      *pkdra=kdra;
      return;
      }

//#include<stdio.h>
//#include<math.h>

//#include<alloc.h>

      void wiczn(int za,int zb,int *pzc,
       int n,int nd,int *a,int *ad,
      int m,int md,int *b,int *bd,
      int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr)
      {
 extern void sl(int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr);

extern void wich(int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr);

// c-длинное вычитание с учетом знаков уменьшаемого и вычитаемого
 int c[401],cd[401],kcel,kdro,zc,
 ar[401],adr[401],
 ai[401],adi[401],bi[401],bdi[401],zais,zbi,ip,
 zbis,i,ni,ndi,mi,mdi,nmin,nr,ndr;

      zais=za;
      zbis=zb;
      for(i=1;i<=n;i++)
      ai[i]=a[i];
      for(i=1;i<=nd;i++)
      adi[i]=ad[i];
      for(i=1;i<=m;i++)
      bi[i]=b[i];
      for(i=1;i<=md;i++)
      bdi[i]=bd[i];
      ni=n;
      ndi=nd;
      mi=m;
      mdi=md;
      zbi=zb;
      zb=zb*(-1);
      ip=za*zb;
      if(ip>0) goto m781;
      if(za==1) goto m782;
m1901:if(m>n) goto m783;
      if(m==n) goto m784;
      wich(n,nd,a,ad,m,md,b,bd,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=za;
      goto m785;
m784:;
      for(i=1;i<=m;i++)
      {
      if(a[i]>b[i]) goto m787;
      if(a[i]<b[i]) goto m788;
      }
      nmin=nd;
      if(nmin>md) nmin=md;
      for(i=1;i<=nmin;i++)
```

```
      {
      if(ad[i]>bd[i]) goto m792;
      if(ad[i]<bd[i]) goto m793;
      }
      if(nd<=md) goto m795;
      wich(n,nd,a,ad,m,md,b,bd,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=za;
      goto m785;
m795:;
      wich(m,md,b,bd,n,nd,a,ad,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=zb;
      goto m785;
m793:;
      wich(m,md,b,bd,n,nd,a,ad,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=zb;
      goto m785;
m792:;
      wich(n,nd,a,ad,m,md,b,bd,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=za;
      goto m785;
m788:;
      wich(m,md,b,bd,n,nd,a,ad,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=zb;
      goto m785;
m787:;
      wich(n,nd,a,ad,m,md,b,bd,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=za;
      goto m785;
m783:;
      wich(m,md,b,bd,n,nd,a,ad,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=zb;
      goto m785;
m782:;
      for(i=1;i<=n;i++)
      ar[i]=a[i];
      for(i=1;i<=nd;i++)
      adr[i]=ad[i];
      nr=n;
      ndr=nd;
      for(i=1;i<=m;i++)
      a[i]=b[i];
      for(i=1;i<=md;i++)
      ad[i]=bd[i];
      n=m;
      nd=md;
      m=nr;
      md=ndr;
      for(i=1;i<=nr;i++)
      b[i]=ar[i];
      for(i=1;i<=ndr;i++)
      bd[i]=adr[i];
      zc=za;
      za=zb;
      zb=zc;
      goto m1901;
m781:;
      sl(n,nd,a,ad,m,md,b,bd,&kcel,&kdro,rezc,rezdr,krazr);
      *pzc=za;
m785:;
      zb=zbi;
      n=ni;
      nd=ndi;
      m=mi;
```

```
      md=mdi;
      for(i=1;i<=n;i++)
      a[i]=ai[i];
      for(i=1;i<=nd;i++)
      ad[i]=adi[i];
      for(i=1;i<=m;i++)
      b[i]=bi[i];
      for(i=1;i<=md;i++)
      bd[i]=bdi[i];
      za=zais;
      zb=zbis;
      *pkcel=kcel;
      *pkdro=kdro;
      return;
      }

//#include<stdio.h>
//#include<math.h>
//#include<alloc.h>
      void ymnzn(int za,int zb,int *pzc,int n,int nd,int *a,int *ad,
      int m,int md,int *b,int *bd,
      int *pkcel,int *pkdro,int *ctot,int *cdtot,int krazr)
      {

 extern void ymn(int n,int nd,int *a,int *ad,
int m,int md,int *b,int *bd,
int *pkcel,int *pkdro,int *ctot,int *cdtot,int krazr);


//c-длинное умножение с учетом знаков сомножителей
      int c[400],cd[400],kcel,kdro,zc,
            csl[400],csld[400],
       rezc[400],rezdr[400],
      ai[400],adi[400],bi[400],bdi[400];
      int zais,zbis,i,ni,ndi,mi,mdi;


      zais=za;
      zbis=zb;
      for(i=1;i<=n;i++)
      ai[i]=a[i];
      for(i=1;i<=nd;i++)

      adi[i]=ad[i];
      for(i=1;i<=m;i++)
      bi[i]=b[i];
      for(i=1;i<=md;i++)
      bdi[i]=bd[i];
      ni=n;
      ndi=nd;
      mi=m;
      mdi=md;

      *pzc=za*zb;

      ymn(n,nd,a,ad,m,md,b,bd,pkcel,pkdro,ctot,cdtot,krazr);

      n=ni;
      nd=ndi;
      m=mi;
      md=mdi;
      for(i=1;i<=n;i++)
      a[i]=ai[i];
```

```
      for(i=1;i<=nd;i++)
      ad[i]=adi[i];
      for(i=1;i<=m;i++)
      b[i]=bi[i];
      for(i=1;i<=md;i++)
      bd[i]=bdi[i];
      za=zais;
      zb=zbis;

      return;

      }
//c-длинное умножение ymn

      void ymn(int n,int nd,int *a,int *ad,
      int m,int md,int *b,int *bd,
      int *pkcel,int *pkdro,int *ctot,int *cdtot,int krazr)
      {

extern void sl(int icel,int idr,int *csl,int *csld,
int ntot,int kdtot,int *ctot,int *cdtot,
int *pkcel,int *pkdro,int *rezc,int *rezdr,int krazr);


      int c[400],cd[400],i,it,in,ik,ntot,kdtot,im,p,nt,mt,jp,pr,i1,os,
      kraz,kcd,icel,iab,idr,j,il,kcel,kdro,
      csl[400],csld[400],
      rezc[400],rezdr[400];
      for(i=1;i<=399;i++)
      {
      ctot[i]=0;
      cdtot[i]=0;
      }
//c-формирование двух массивов для умножения
      it=0;
      in=n+1;
      ik=n+nd;
      for(i=in;i<=ik;i++)
      {
      it=it+1;
      a[i]=ad[it];
      }
      it=0;
      in=m+1;
      ik=m+md;
      for(i=in;i<=ik;i++)
      {
      it=it+1;
      b[i]=bd[it];
      }
      ntot=n+m;
      kdtot=nd+md;
      for(i=1;i<=ntot;i++)
      ctot[i]=0;
      for(i=1;i<=kdtot;i++)
      cdtot[i]=0;

//c-умножение
      i=0;
      im=0;
      p=0;
      nt=n+nd;
      mt=m+md;
```

```
m801:;

      p=0;
      for(jp=1;jp<=399;jp++)
      c[jp]=0;
      i=0;
      nt=n+nd;
      im=im+1;
m701:;
      pr=a[nt]*b[mt]+p;
      i1=pr/10;
      os=pr-i1*10;
      i=i+1;
      c[i]=os;
      p=i1;
       nt=nt-1;
      if(nt>=1) goto m701;
      if(p==0) goto m1701;
      i=i+1;
      c[i]=p;
      p=0;
m1701:;

      kraz=md-im+1;
      kcd=nd+kraz;
      icel=i-kcd;
      if(icel<i) goto m1921;
      iab=kcd;
      if(kcd<0)  iab=-kcd;
      it=0;
      for(jp=i;jp>=1;jp=jp-1)

      {
      it=it+1;
      csl[it]=c[jp];
      }
      if(iab==0) goto m1941;
      for(jp=iab;jp>=1;jp=jp-1)
      {
      it=it+1;
      csl[it]=0;
      }
m1941:;
      idr=1;
      csld[1]=0;
      goto m8722;
m1921:;
      it=0;
      in=i;
      ik=kcd+1;
      if(in==ik) goto m961;
      if(in<ik) goto m962;
      for(j=in;j>=ik;j=j-1)
      {
      it=it+1;
      csl[it]=c[j];
      }
      icel=i-kcd;
      goto m963;
m961:;
      icel=1;
      csl[1]=c[in];
      goto m963;
```

```
m962:;
      icel=1;
      csl[1]=0;
m963:;
      in=kcd;
      it=0;
      for(j=in;j>=1;j=j-1)
      {
      it=it+1;
      csld[it]=c[j];
      }
      idr=kcd;
      if(idr>0) goto m8722;
      idr=1;
      csld[1]=0;
m8722:;

      sl(icel,idr,csl,csld,ntot,kdtot,ctot,cdtot,&kcel,&kdro,
     rezc,rezdr,krazr);
      if(kcel!=0) goto m8723;
      kcel=1;
      rezc[1]=0;
m8723:;
      if(kdro!=0) goto m8724;
      kdro=1;
      rezdr[1]=0;
m8724:;
      for(i=1;i<=kcel;i++)
      ctot[i]=rezc[i];
      for(i=1;i<=kdro;i++)
      cdtot[i]=rezdr[i];
      ntot=kcel;
      kdtot=kdro;
      mt=mt-1;
      if(mt>=1) goto m801;
      it=0;
      for(j=1;j<=kcel;j++)
      {
      if(ctot[j]!=0) goto m7767;
      it=it+1;
      }
m7767:;
      il=0;
      for(jp=j;jp<=kcel;jp++)
      {
      il=il+1;
      ctot[il]=ctot[jp];
      }
      kcel=kcel-it;
      if(kcel!=0) goto m3999;
      kcel=1;
      ctot[1]=0;
m3999:;
      if(kdro!=0) goto m4000;
      kdro=1;
      cdtot[1]=0;
m4000:;
      if(kdro>krazr) kdro=krazr;
      *pkcel=kcel;
      *pkdro=kdro;
      return;

      }
```

Приложение-4(выходные файлы с результатами)
Файл ch_pi(программа chispi.for,pikontr.for)

```
 шаг интегрирования

 целая часть шага
 0

 дробная часть шага
 00001

 число пи

 целая часть
 3

 дробная часть
 14160265357312457160
```

Файл pipardl(программа pidlpar1.for)

```
 количество разрядов в шаге dx=          5
 количество разрядов в дробной части=   20
 количество процессов=14


 шаг интегрирования

 целая часть шага
 0

 дробная часть шага
 00001

 число пи

 целая часть
 3

 дробная часть
 14160265357312457160
```

Файл pidlp_c(программа pidlpar.c)

```
количество разрядов в шаге dx=5
количество разрядов в дробной части=20
количество процессов=12
шаг интегрирования
целая часть шага
0
дробная часть шага
00001
число пи
целая часть
3
дробная часть
14160265357312457160
```

Протокол с кластера вычисления числа пи

```
Start  date:Ё╬─ э└╥ 31 10:24:27 YEKST 2008
количество разрядов в шаге dx=5
количество разрядов в дробной части=20
количество процессов=3
шаг интегрирования
целая часть шага
0
дробная часть шага
00001
число пи
целая часть
3
дробная часть
14160265357312457160
 End  date:Ё╬─ э└╥ 31 10:25:02 YEKST 2008
```