Нисходящее проектирование алгоритмов и программ

Варшавский Павел Романович

к.т.н., доцент кафедры ПМ

Список литературы по курсу

1. Князев А.В. Основы языка C++. Учебное пособие. –М.: Издательство МЭИ, 2013 – 80 с. ISBN 978-5-7046-1425-8.

2. Князев А.В. Работа со сложными структурами данных на языке C++. Учебное пособие. –М.: Издательство МЭИ, 2015 – 48 с. ISBN 978-5-7046-1658-0.

3. Программирование. Сборник задач. Учебное пособие. –Санкт-Петербург: Лань, 2019 – 140 с. ISBN 978-5-8114-3857-0.

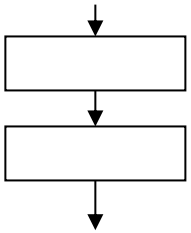
URL: https://e.lanbook.com/book/121485

ПРИНЦИПЫ СТРУКТУРНОГО ПРОГРАММИРОВАНИЯ

НИСХОДЯЩЕЕ ПРОЕКТИРОВАНИЕ

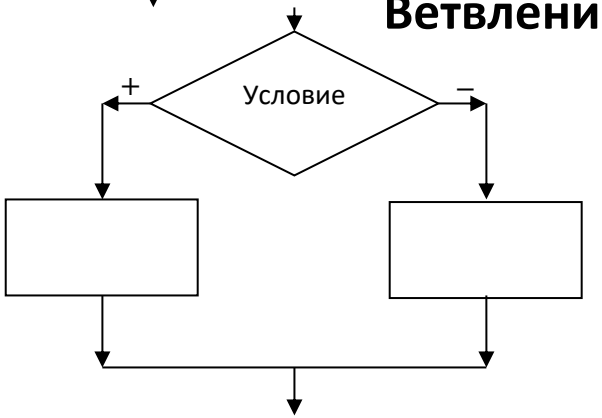
Структурное программирование – парадигма программирования, в основе которой лежит представление программы в виде иерархической структуры блоков, использующих базовые управляющие алгоритмические структуры (набор из **следования**, **ветвления** и **цикла «пока»** называется ***базисом Дейкстры***, а набор из **следования**, **ветвления** и **цикла «до»** называется ***базисом Вирта***).

Базис Дейкстры

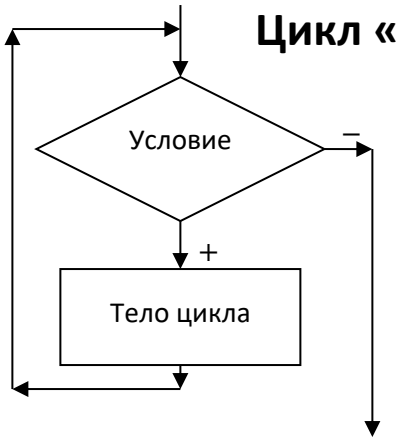


Следование

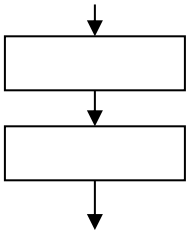
Ветвление



Цикл «пока»

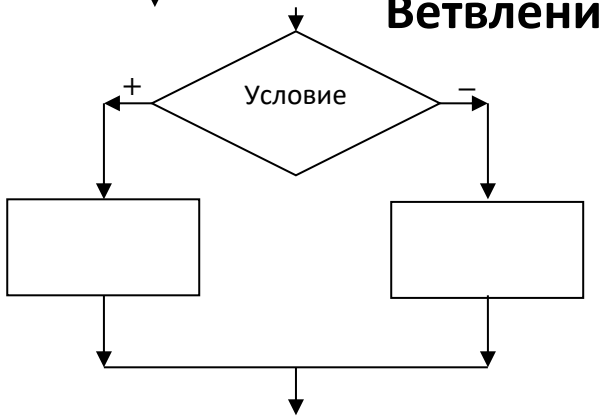


Базис Вирта

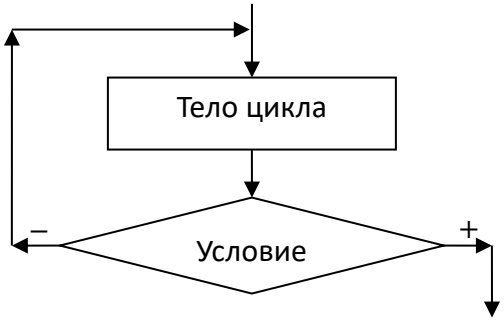


Следование

Ветвление



Цикл «до»



Методология структурного программирования была разработана **Э. Дейкстрой** и **Н. Виртом** в конце 1960 — начале 1970 годов из-за возрастания сложности решаемых на компьютерах задач и проблем, связанных с программами, в которых используется оператор безусловного перехода.

Её основой стала **теорема Бёма-Якопини** (1965 год), что применение безусловного перехода не является обязательным, т.е. не существует такой программы с безусловным переходом, которую нельзя было бы записать без него с полным сохранением функциональности.

Принципы структурного программирования

- Программа (алгоритм) должна разделяться на независимые части (блоки, подпрограммы).
- Подпрограмма имеет одну входную и одну выходную точку (в отличие от программ с использованием оператора безусловного перехода).
- Используются только три базовые управляющие конструкции (следование, ветвление, цикл).

Принципы структурного программирования

- В программе базовые управляющие конструкции могут быть вложены друг в друга (без их пересечения).
- Повторяющиеся фрагменты программы можно оформить в виде подпрограмм (процедур или функций).
- Разработка программы ведётся пошагово, методом «сверху вниз» (нисходящее проектирование).

Достоинства структурного программирования

- повышается надежность программ;
- повышается эффективность программ;
- уменьшается время и стоимость программной разработки;
- улучшается читабельность программ.

НИСХОДЯЩЕЕ ПРОЕКТИРОВАНИЕ

Для построения структурированных алгоритмов в рамках методологии структурного программирования была разработана специальная технология – **нисходящее проектирование**, которая состоит в пошаговой детализации (декомпозиции, разложении) задачи на подзадачи.

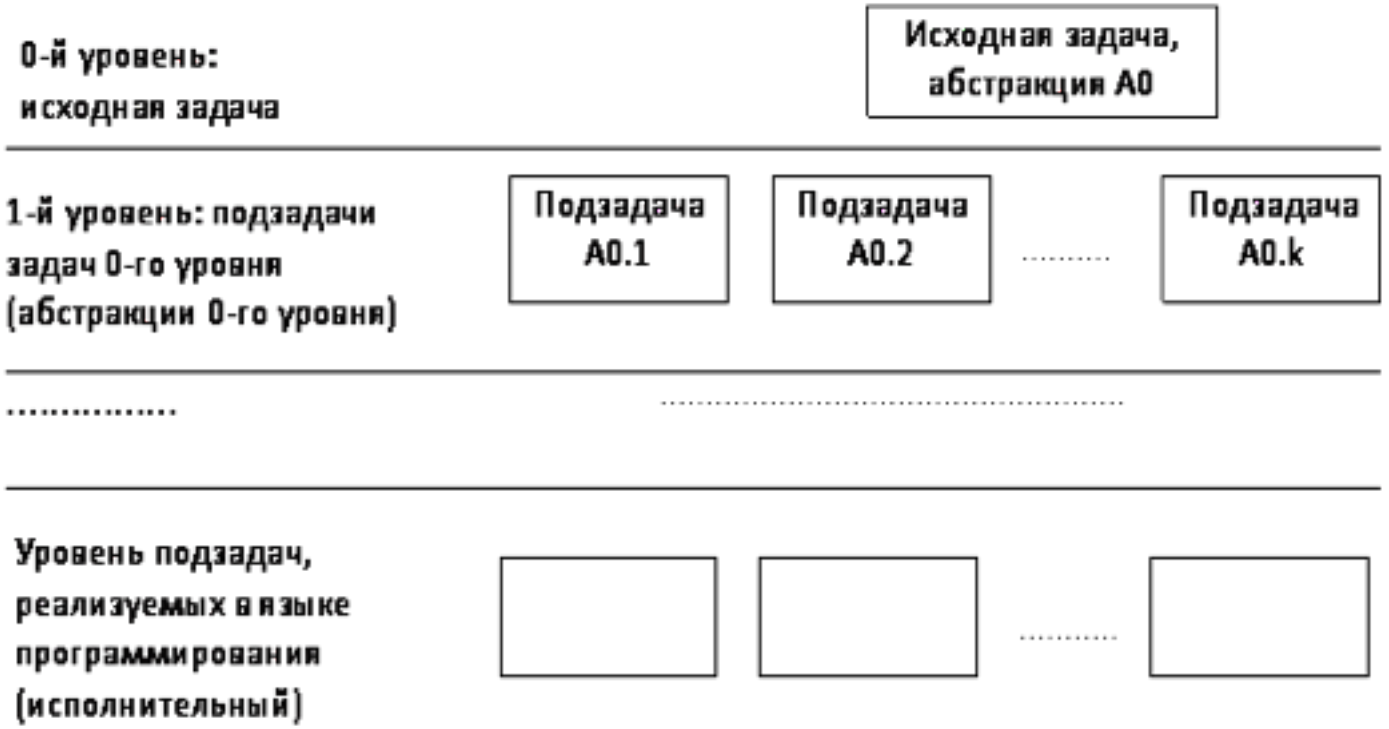
НИСХОДЯЩЕЕ ПРОЕКТИРОВАНИЕ

Пошаговая детализация представляет собой процесс дробления задачи на подзадачи (абстракции различного уровня), установления логических связей между ними.

После этого переходят к уточнению выделенных подзадач.

Этот процесс детализации продолжается до уровня, позволяющего достаточно легко реализовать подзадачу на выбранном языке программирования.

Общая схема нисходящей разработки



При пошаговой детализации используется **принцип замещения**, который состоит в замене любого функционального блока (прямоугольника) блок-схемы некоторой базовой структурой (следование, ветвление, повторение).

ВОСХОДЯЩЕЕ ПРОЕКТИРОВАНИЕ

Восходящее проектирование – методика разработки программ, при которой крупные блоки собираются из ранее созданных мелких блоков.

Случаи восходящего проектирования:

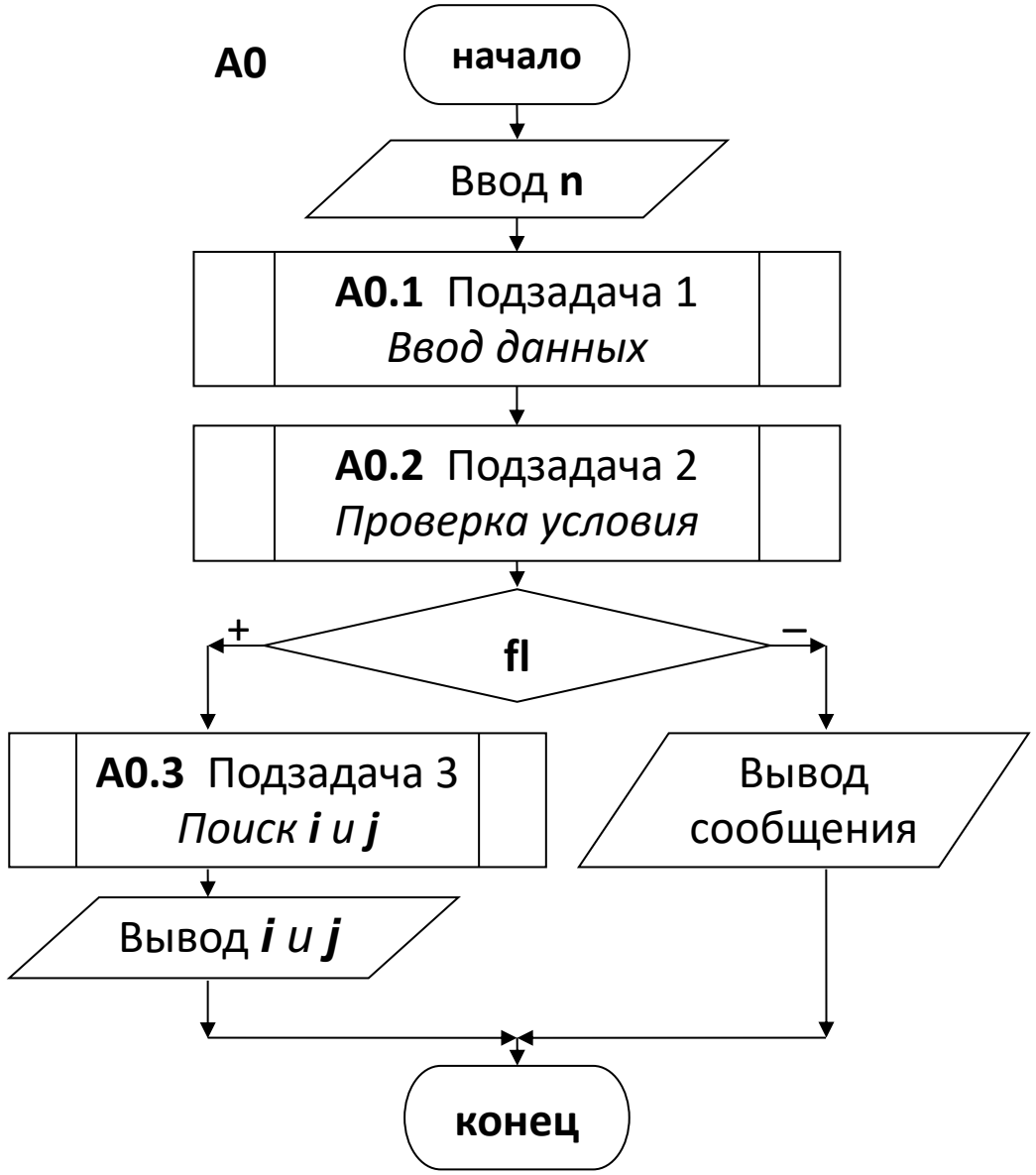
- разработчик ясно представляет направление поиска, но не знает заранее, как далеко он сможет продвинуться к цели;
- нет возможности предвидеть объем ресурсов для достижения того или иного результата;
- разработка не поддается детальному планированию, она ведется методом проб и ошибок.

ПРИМЕР (задача 5.2 вариант №1)

Даны два массива $C_1, C_2, \ldots, C_n$ и $P_1, P_2, \ldots, P_n$. Если каждый элемент первого массива меньше суммы элементов второго, найти, при каких значениях i и j максимально значение выражения $C_i / (P_j + C_i^2)$.

Таблица данных основной программы

Имя	Смысл	Тип	Структура
	Входные данные		
n	размерность массивов	цел	простая переменная
C	первый массив	вещ	одномерный массив
P	второй массив	вещ	одномерный массив
	Выходные данные		
i	параметр в выражении	цел	простая переменная
j	параметр в выражении	цел	простая переменная
	Промежуточные данные		
fl	условие по задаче	лог	простая переменная



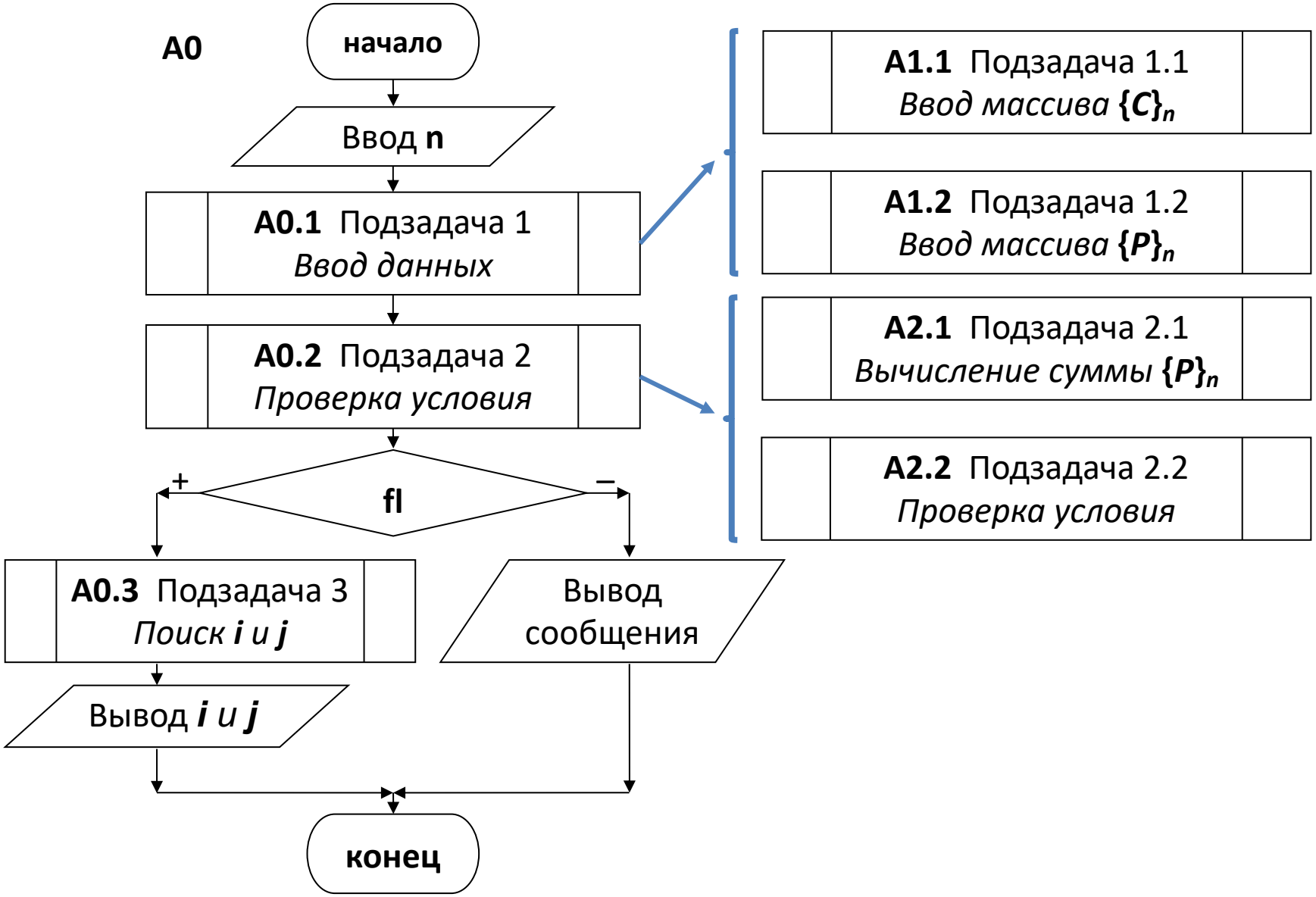
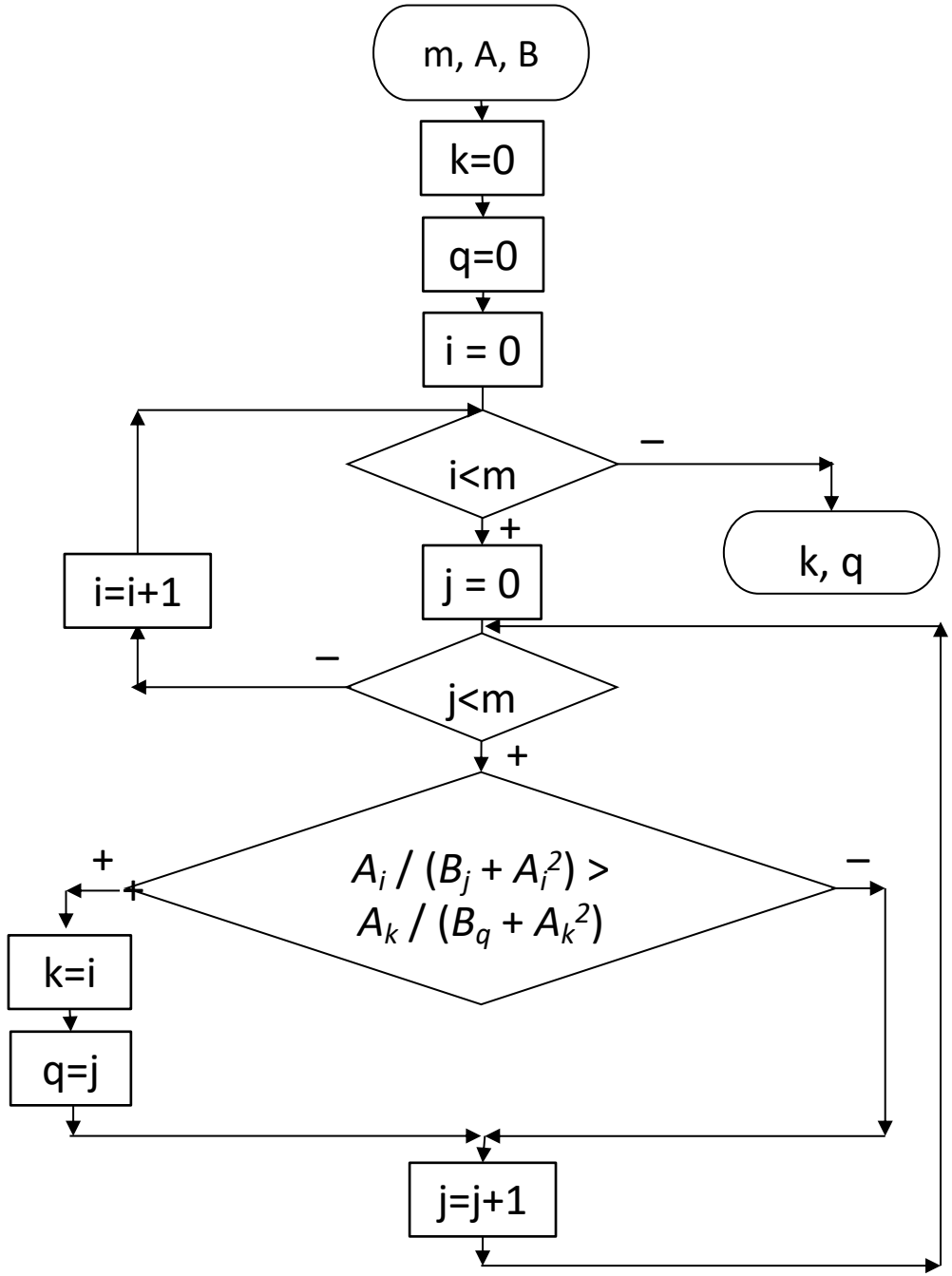

A0
начало
Ввод n
A0.1 Подзадача 1
Ввод данных
A0.2 Подзадача 2
Проверка условия
+
fl
–
A0.3 Подзадача 3
Поиск i и j
Вывод i и j
Вывод
сообщения
конец
A1.1 Подзадача 1.1
Ввод массива {C}n
A1.2 Подзадача 1.2
Ввод массива {P}n
A2.1 Подзадача 2.1
Вычисление суммы {P}n
A2.2 Подзадача 2.2
Проверка условия

Таблица данных
(А0.3 Подзадача 3)

Имя	Смысл	Тип	Структура
	Входные данные		
m	размерность массивов	цел	простая переменная
A	первый массив	вещ	одномерный массив
B	второй массив	вещ	одномерный массив
	Выходные данные		
k	параметр в выражении	цел	простая переменная
q	параметр в выражении	цел	простая переменная
	Промежуточные данные		
i,j	параметры цикла	цел	простая переменная



```
void InputMas (int m, double A[]);
double SumP (int m, double B[]);
void Check (int m, double A[], double B[], bool &f);
void Findij (int m, double A[], double B[], int &k, int &q);

void main ()
{
    int n, i, j;
    double C[10], P[10];
    bool fl;
    cout<<“Введите количество элементов в массивах”;
    cin>>n;
    InputMas (n, C);          // Подзадача 1.1
    InputMas (n, P);          // Подзадача 1.2
    Check (n, C, P, fl);      // Подзадача 2
    if (fl==true)
    {
        Findij (n, C, P, i, j);          // Подзадача 3
        cout<<i<<j<<endl;
    }
    else   cout<<“Условие не выполнено”<<endl;
}
```

```
void Findij (int m, double A[], double B[], int &k, int &q)
{
    int i, j;
    k=0;
    q=0;
    for (i=0;i<m;i++)
        for (j=0;j<m;j++)
            if ((A[i]/(B[j] + A[i]*A[i]) > (A[k]/(B[q] + A[k]*A[k]))
            {
                k=i;
                q=j;
            }
}
```