Программирование и основы алгоритмизации

К.т.н., доцент
Гаврилов Андрей Владимирович
http://www.insycom.ru
Email: andr_gavrilov@yahoo.com
V-241

Цели курса

- Дальнейшее изучение программирования на языке С
  - Работа с файлами
  - Структуры данных
  - Списки, работа с указателями
  - Алгоритмы сортировки и поиска
- Технологии программирования
  - Структурное
  - Нисходящее
  - Визуальное
  - Объектно-ориентированное
- Основы программирования на С++ (CBuilder)
- Введение в вычислительные методы

Аттестация

Соответствие баллов (кредитов) видам учебной деятельности

Виды учебной деятельности	Баллы (максимально возможное количество)	Сроки сдачи
Лекции (34)	10 (посещение всех лекций)	в соответствии с расписанием
Лабораторная работа (8 занятий)	5 (40)	в соответствии с расписанием
Расчетно-графическое задание(1)	30	РГЗ – до 17 недели
Зачет	20	Зачетная неделя

<table>
<tr><th>Характеристика работы студента</th><th>Диапазон баллов рейтинга</th><th>Оценка ECTS</th><th colspan="2">Традиционная (4-уровневая) шкала оценки</th></tr>
<tr><td rowspan="3">«Отлично»</td><td rowspan="3">90-100</td><td>A+</td><td rowspan="3">отлично</td><td rowspan="7">зачтено</td></tr>
<tr><td>A</td></tr>
<tr><td>A-</td></tr>
<tr><td rowspan="3">«Очень хорошо»</td><td rowspan="3">80-89</td><td>B+</td></tr>
<tr><td>B</td><td rowspan="3">хорошо</td></tr>
<tr><td>B-</td></tr>
<tr><td rowspan="3">«Хорошо»</td><td rowspan="3">70-79</td><td>C+</td></tr>
<tr><td>C</td></tr>
<tr><td>C-</td><td>удовлетворительно</td></tr>
<tr><td rowspan="3">«Удовлетворительно»</td><td rowspan="3">60-69</td><td>D+</td><td rowspan="4">удовлетворительно</td><td rowspan="4">зачтено</td></tr>
<tr><td>D</td></tr>
<tr><td>D-</td></tr>
<tr><td>«Посредственно»</td><td>50-59</td><td>E</td></tr>
<tr><td>«Неудовлетворительно»</td><td>25-49</td><td>FX</td><td rowspan="2">неудовлетво-рительно</td><td rowspan="2">не зачтено</td></tr>
<tr><td>«Неудовлетворительно»</td><td>0-24</td><td>F</td></tr>
</table>

Литература

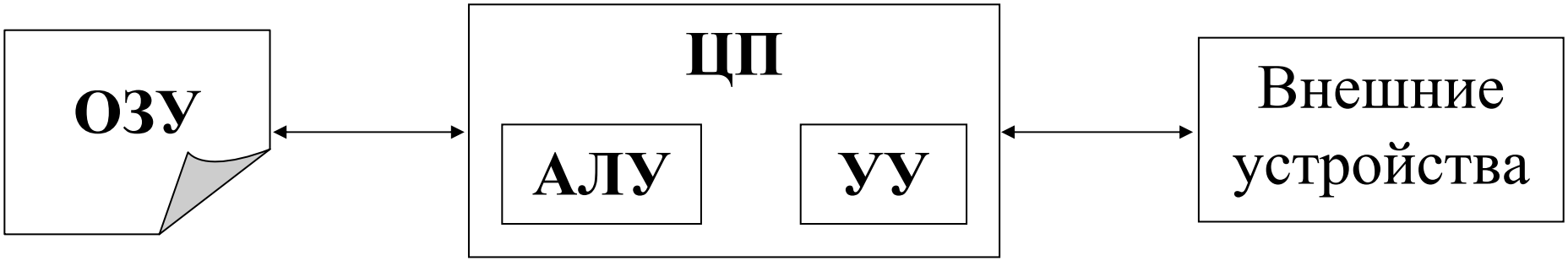
- Программирование на языке С.(.chm)
- В.Алексанкин, Н. Елманова. Начинаем работать в Borland C++ Builder (doc)
- C++ Builder. Обучалка. 7 уроков. (.HTML)
- Лекции по языку С. 8 лекций. (.Doc)
- К.Рейсдорф, К.Хендерсон. Borland C++Builder. Освой самостоятельно. (Pdf)
- Т.А.Павловская, Ю.А.Щупак. С++. Объектно-ориентированное программирование. Практикум. Уч. пособие. Питер, 2006. (pdf)
- Н.Культин. C++ Builder в задачах и примерах. С.-Петербург: БХВ-Петербург, 2005. (pdf)
- Липпман. С++ для начинающих. (pdf)
- В.В.Мухортов. В.Ю. Рылов. Объектно-ориентированное программирование. Анализ и дизайн. Методическое пособие. Новосибирск: ООО «Новософт», 2002.
- Д.Хенкеманс, М.Ли. Программирование на С++. С.-Петербург: Символ, 2002. (pdf)
- Алгоритмы, методы, исходники. (.chm)
- Е.Л.Романов. Язык Си. Типы данных и управление памятью. Конспект лекций. Новосибирск: НГТУ, 2000.
- Е.Л.Романов. C/C++/ От дилетанта до профессионала
  http://ermak.cs.nstu.ru/cprog/HTML/index.htm

Лекция 1

Повторение основ программирования и алгоритмизации из курса «Информатика»

Компьютер фон Неймана

Структура, основные компоненты компьютера фон Неймана



ОЗУ – оперативное запоминающее устройство,
АЛУ – арифметико-логическое устройство,
УУ – устройство управления, ЦП – центральный процессор

Принципы построения компьютера фон Неймана

1. Принцип двоичного кодирования
2. Принцип программного управления
3. Принцип хранимой программы

Центральный процессор

Рабочий цикл процессора

Виртуальная память

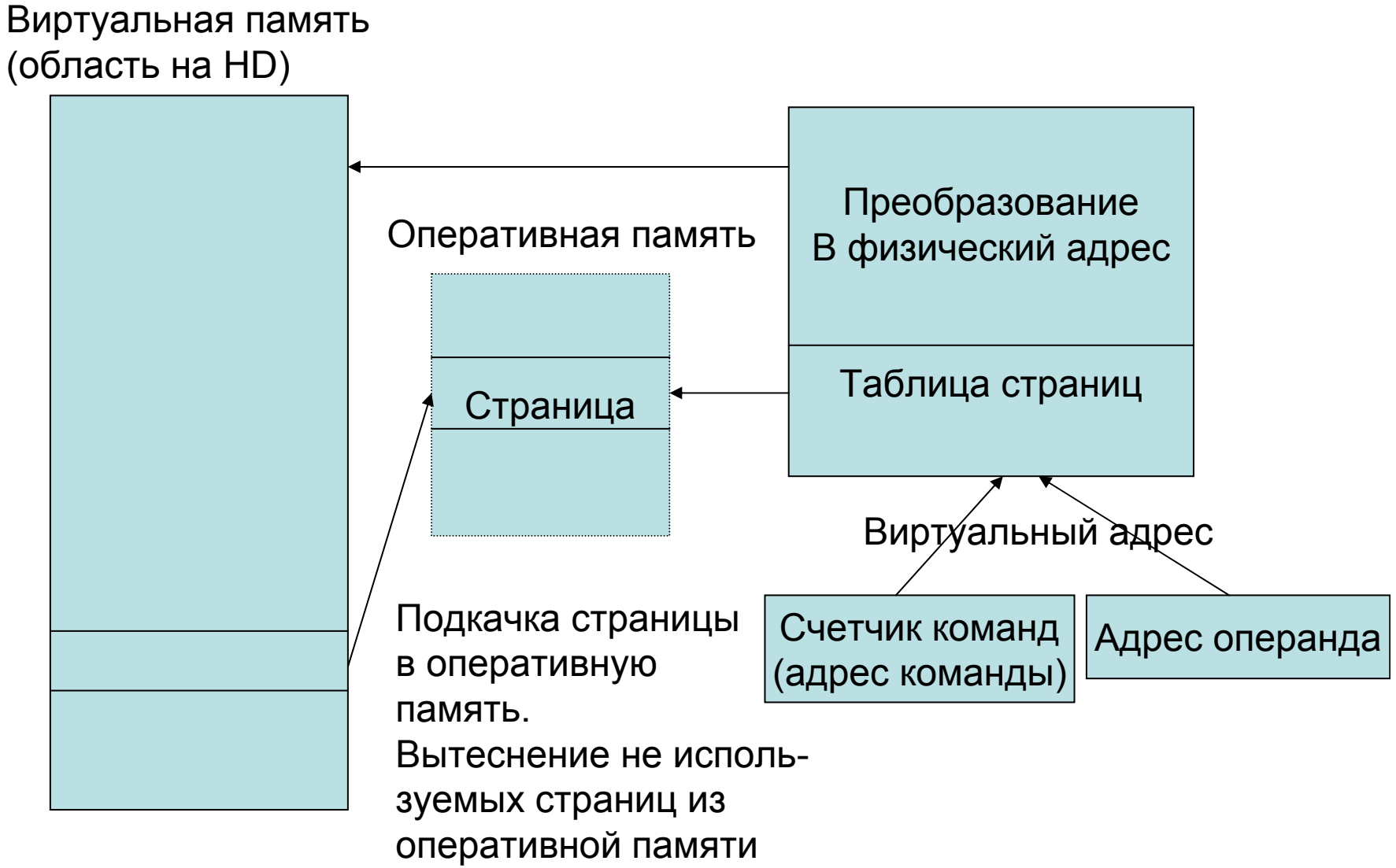
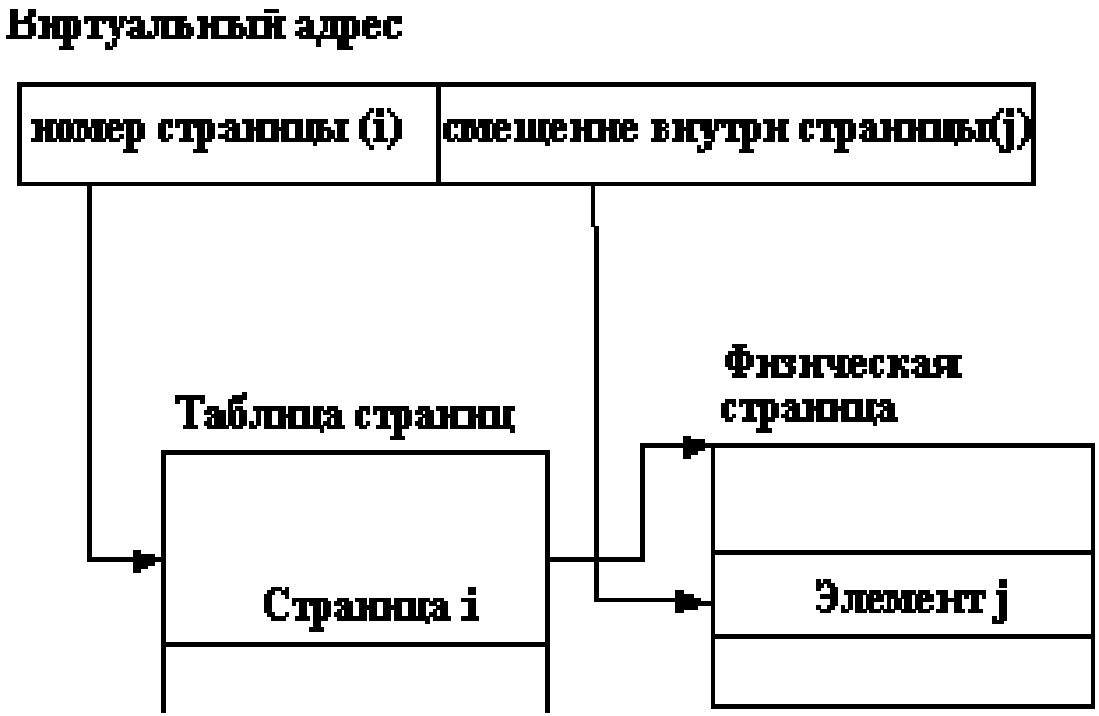


Схема организации виртуальной памяти



Основные этапы решения задач с помощью компьютера:

- Постановка задачи;
- Анализ и исследование задачи, модели;
- Разработка алгоритма;

- Программирование (кодирование);
- Тестирование и отладка;
- Анализ результатов решения задачи;
- Сопровождение программы.

Определения алгоритма

- «Алгоритм — это конечный набор правил, который определяет последовательность операций для решения конкретного множества задач и обладает пятью важными чертами: конечность, определённость, ввод, вывод, эффективность». (Д. Э. Кнут)
- «Алгоритм — это всякая система вычислений, выполняемых по строго определённым правилам, которая после какого-либо числа шагов заведомо приводит к решению поставленной задачи». (А. Колмогоров)
- «Алгоритм — это точное предписание, определяющее вычислительный процесс, идущий от варьируемых исходных данных к искомому результату». (А. Марков)
- «Алгоритм — строго детерминированная последовательность действий, описывающая процесс преобразования объекта из начального состояния в конечное, записанная с помощью понятных исполнителю команд». (Николай Дмитриевич Угринович)
- «Алгоритм — это последовательность действий, направленных на получение определённого результата за конечное число шагов». (ROXANstudio)
- «Алгоритм — это строго определённая последовательность действий, направленная на достижение определённых целей за конечное число шагов». (Привалов Егор Николаевич)
- «Алгоритм есть формализованная последовательность действий (событий). Алгоритм может быть записан словами и изображён схематически. Практически любое неслучайное повторяемое действие поддаётся описанию через алгоритм».

Свойства алгоритма

- Детерминированность — определённость. В каждый момент времени следующий шаг работы однозначно определяется состоянием системы. Таким образом, алгоритм выдаёт один и тот же результат (ответ) для одних и тех же исходных данных.
- Понятность — алгоритм для исполнителя должен включать только те команды, которые ему (исполнителю) доступны, т.е. которые входят в его систему команд.
- Завершаемость (конечность) — при корректно заданных исходных данных алгоритм должен завершать работу и выдавать результат за конечное число шагов. С другой стороны, вероятностный алгоритм может и никогда не выдать результат, но вероятность этого равна 0.
- Массовость — алгоритм должен быть применим к разным наборам исходных данных.

Виды представления алгоритма

- Графический
  - Блок-схемы
- Текстовый
  - Псевдокод – текст на языке, похожем на естественный
  - Код на языке программирования
- Программа
  - Код в памяти компьютера на языке команд компьютера

Пример алгоритма

```
Если дверь заперта, то:
Вставьте ключ в замок
Поверните ключ
Если дверь все еще заперта, то:
Поверните ключ в другую сторону
Потяните за ручку двери
и т.д.
```

Эта часть кода описывает только открывание двери; здесь даже не проверяется, та ли дверь будет открыта. Если замок заклинило или автомобиль оснащен противоугонной системой, алгоритм открывания двери может быть гораздо сложнее.

Псевдокод

- Это компактный (зачастую неформальный) язык описания алгоритмов, использующий ключевые слова языков программирования, но опускающий несущественные подробности и специфический синтаксис
- Псевдокод обычно опускает детали, несущественные для понимания алгоритма человеком. Такими несущественными деталями могут быть описания переменных, системно-зависимый код и подпрограммы
- Главная цель использования псевдокода — обеспечить понимание алгоритма человеком, сделать описание более воспринимаемым, чем исходный код на языке программирования

Псевдокод 2

- Псевдокод занимает промежуточное место между естественным и формальным языками. С одной стороны, он близок к обычному естественному языку, поэтому алгоритмы могут на нем записываться и читаться как обычный текст. С другой строны, в псевдокоде используются некоторые формальные конструкции и математическая символика, что приближает запись алгоритма к общепринятой математической записи.
- **В псевдокоде не приняты строгие синтаксические правила для записи команд**, присущие формальным языкам, что облегчает запись алгоритма на стадии его проектирования и дает возможность использовать более широкий набор команд, рассчитанный на абстрактного исполнителя.
- Однако в псевдокоде обычно **имеются некоторые конструкции, присущие формальным языкам**, что облегчает переход от записи на псевдокоде к записи алгоритма на формальном языке. В частности, в псевдокоде, так же, как и в формальных языках, есть **служебные слова**, смысл которых определен раз и навсегда. Они выделяются в печатном тексте жирным шрифтом, а в рукописном тексте подчеркиваются.
- Единого или формального определения псевдокода не существует, поэтому возможны различные псевдокоды, отличающиеся набором служебных слов и основных (базовых) конструкций.

Базовые управляющие структуры псевдокода

Название структуры	Псевдокод
присваивание, ввод, вывод	переменная = 0, ввод (переменная), вывод (переменная)
ветвление	если условие то (серия1 иначе серия 2)
цикл ПОКА	пока условие нц серия кц

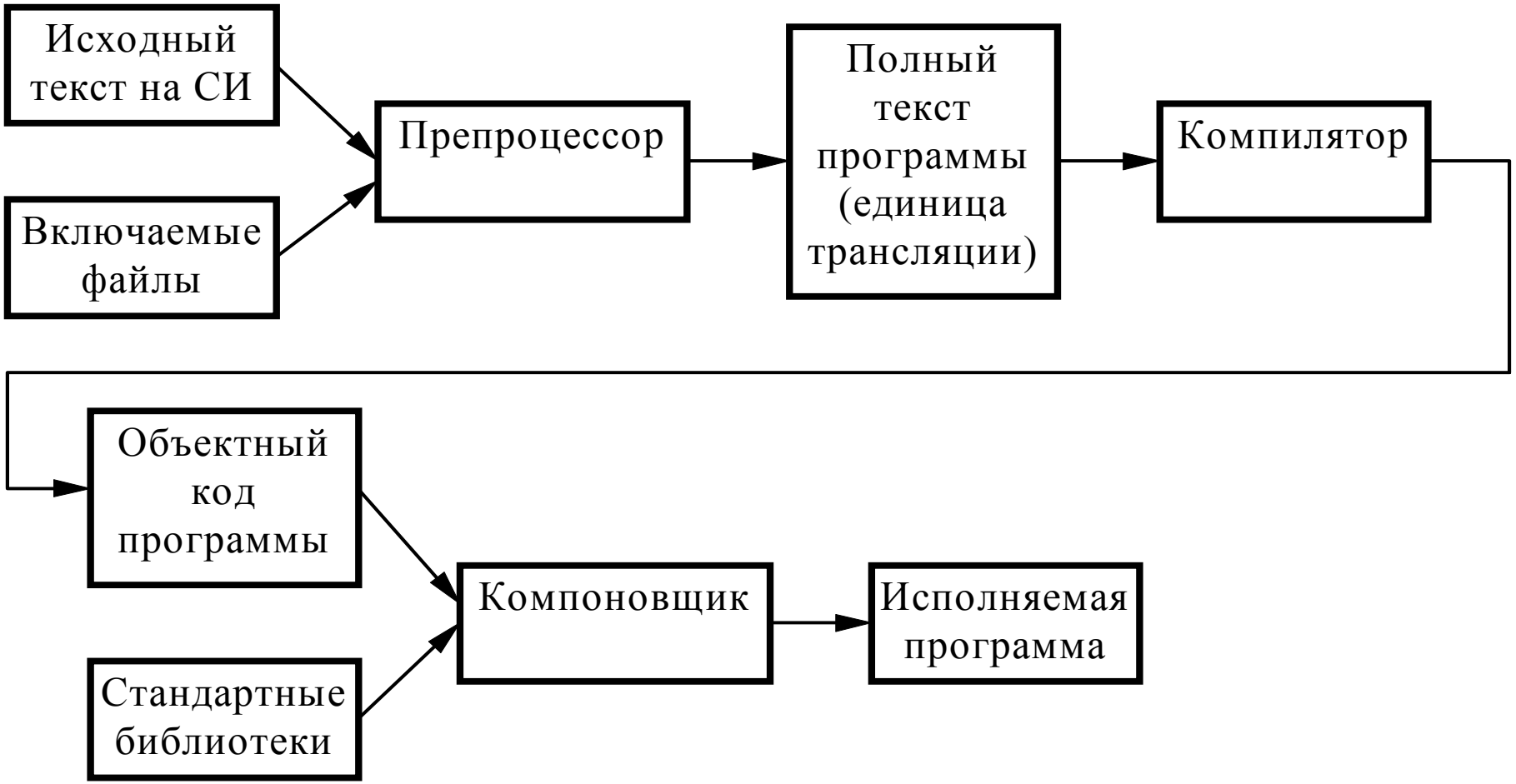
Пример 1 алгоритма «Hello, world» на псевдокоде

```
алг HELLOWORLD
   нач
   вывод ('Hello,World')
   кон алг HELLOWORLD
```

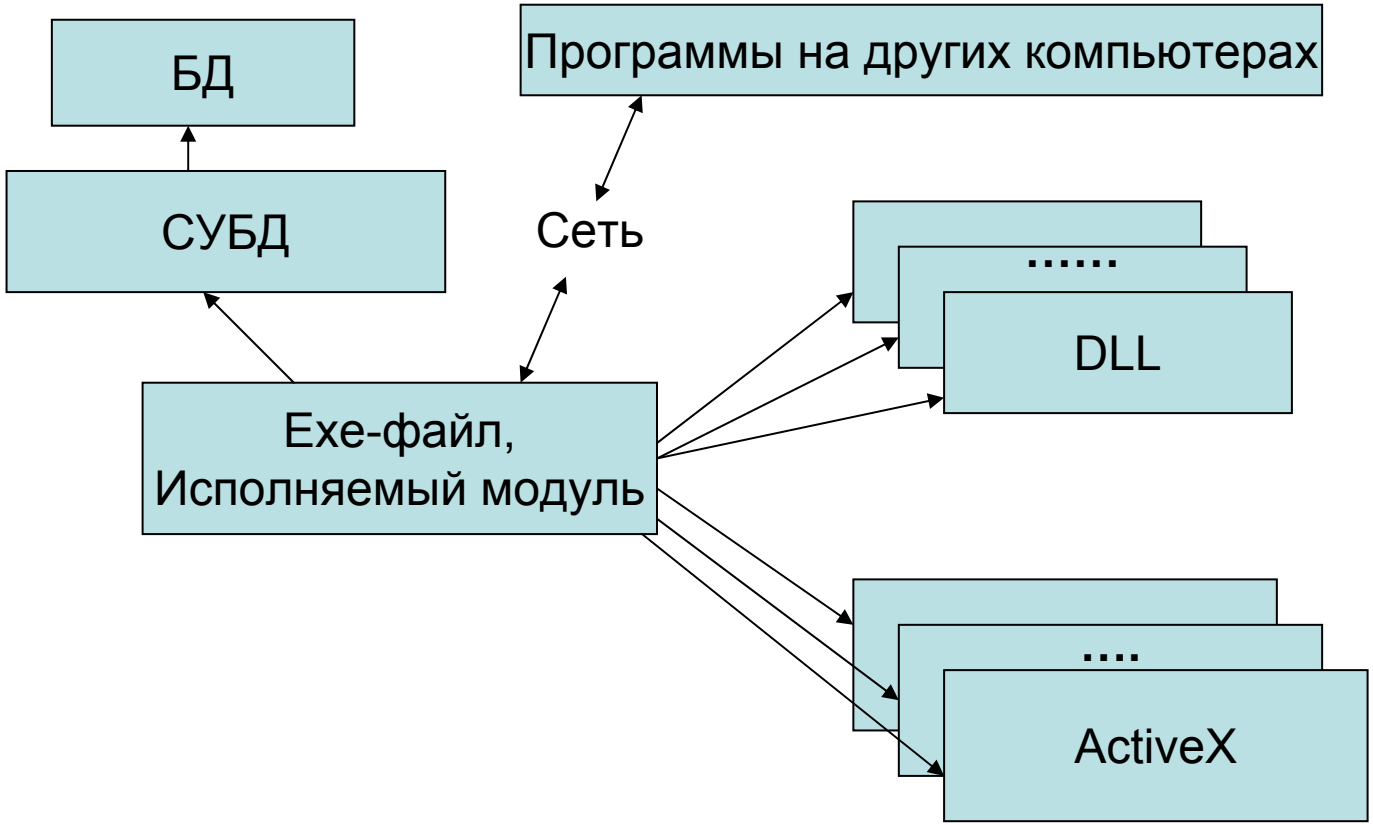
Пример 2 вычисления суммы квадратов первых n натуральных чисел

```
алг Сумма квадратов (арг цел n, рез цел S)
 дано | n > 0
 надо | S = 1*1 + 2*2 + 3*3 + … + n*n
 нач цел i | ввод n; S:=0
   нц для i от 1 до n
        S := S + i * i
   кц
  вывод "S = ", S
 кон
```

Структура и компоненты простой программы на С



Представление о современной сложной программной системе



Ключевые понятия любого алгоритмического языка

- Синтаксис языка
  - Как выглядит исходный текст программы
- Семантика языка
  - Чему соответствует в исполняемой программе то, что написано в исходном тексте
- Переменная:
  - Идентификатор (имя переменной)
  - Тип
  - Область видимости (локальные, глобальные переменные)
- Присваивание
  - Значений переменным
- Оператор – синтаксически законченный фрагмент программы, имеющий определенную семантику
- Связывание
  - Имен переменных и функций с их адресом в виртуальной памяти
- Процедуры и функции (поименованные части программы, которые можно многократно вызывать в программе)

Структура главного модуля программы на языке С

директивы_препроцессора

void main(void)

{

определения_объектов;

исполняемые_операторы;

}

Управляющие операторы

Оператор выражение

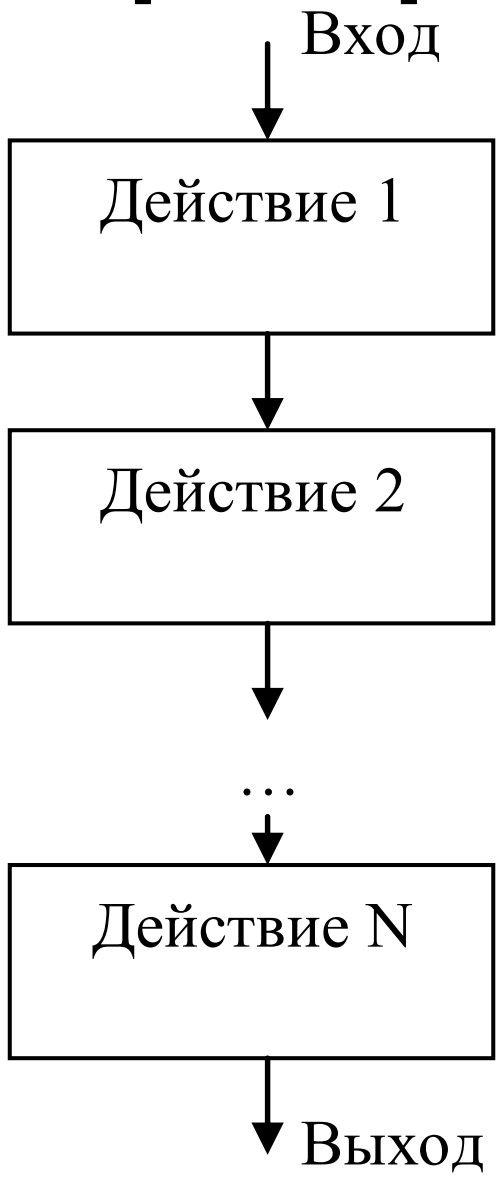
```
++ i;
a=cos(b * 5);
```

Пустой оператор

```
;
```

Составной оператор

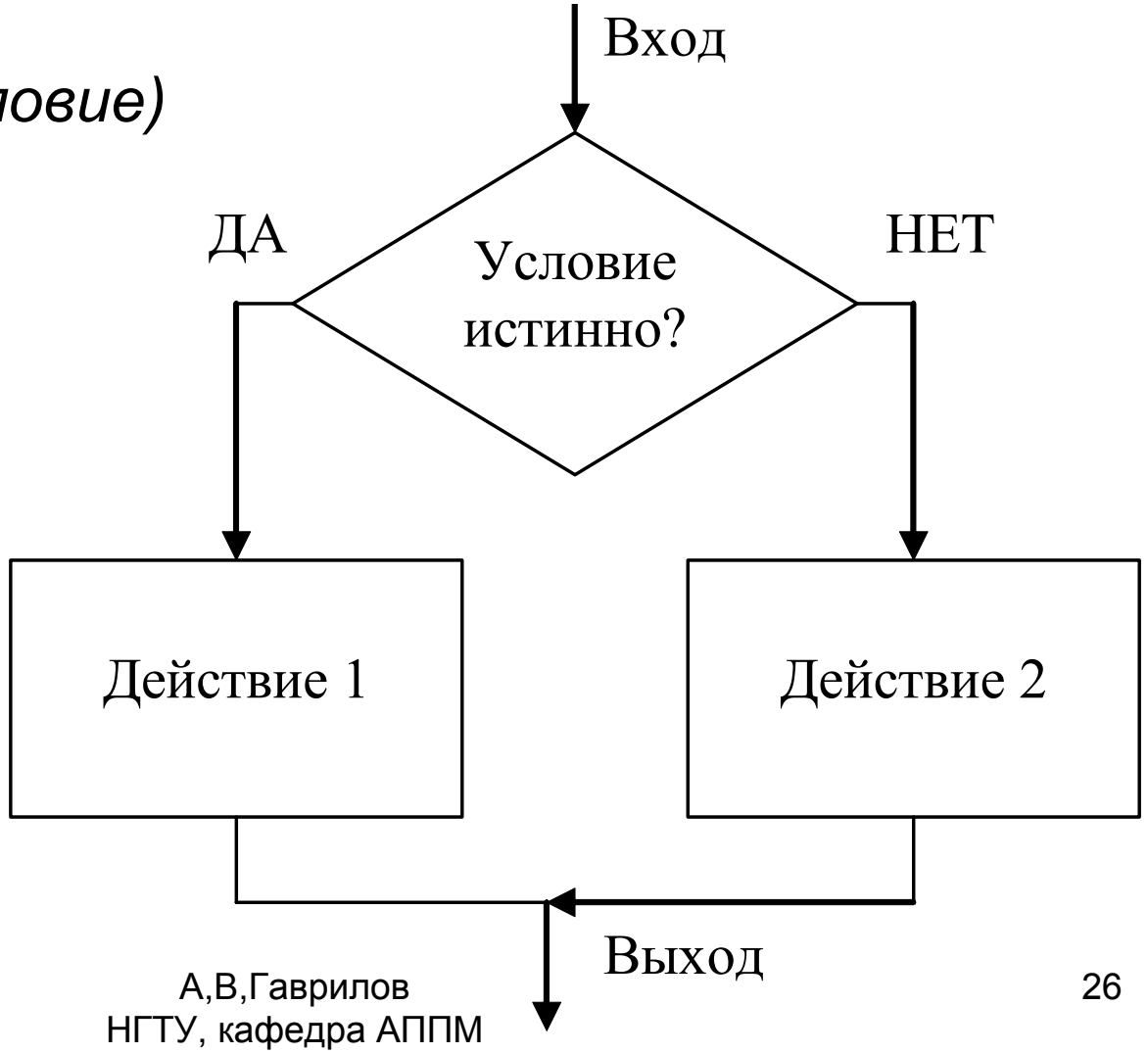
```
{  оператор1;
   оператор2;
    …;
   операторN;
}
```



Условные операторы

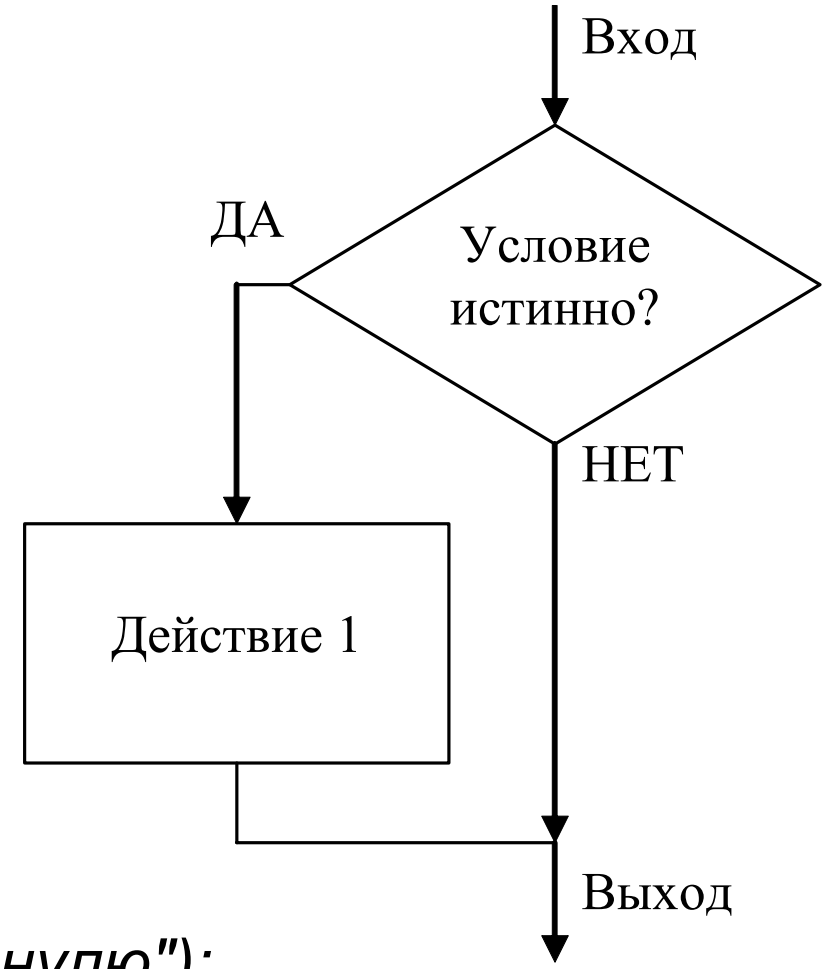
Условный оператор if

if *(выражение_условие)*
оператор_1;
else
оператор_2;



if (выражение_условие***)*** оператор;

if (выражение_условие)
{ оператор_11;
оператор_12;
}
else
{ оператор_21;
оператор_22;
}



Пример1:

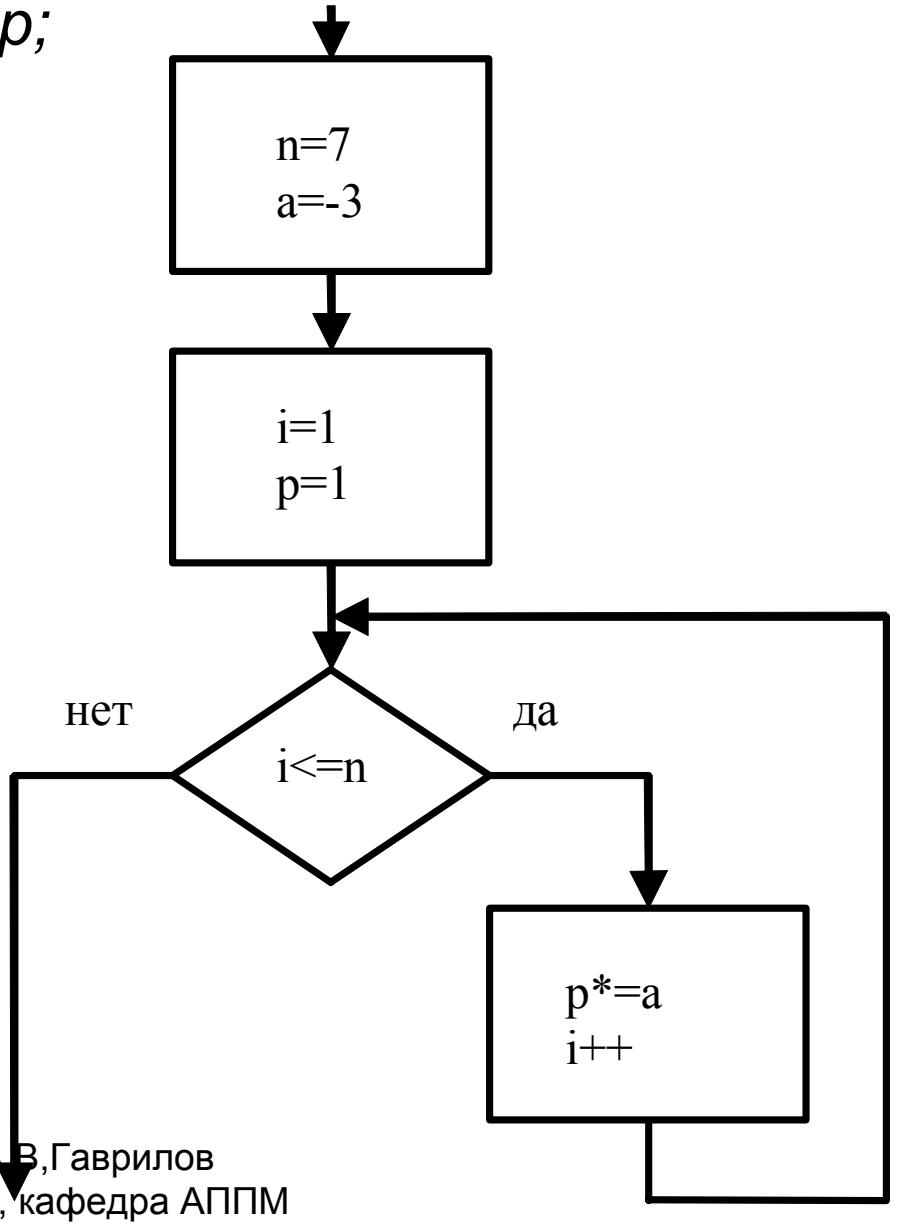
```
if (x) printf("число не равно нулю");
else printf("число равно нулю");
```

Операторы цикла

while *(условие) оператор;*

<u>Пример</u>: Вычислить AN:

```
…
n=7; a=-3;
i=1; p=1;
while (i<=n)
    {
    p*=a;
    i++;
    }
…
```



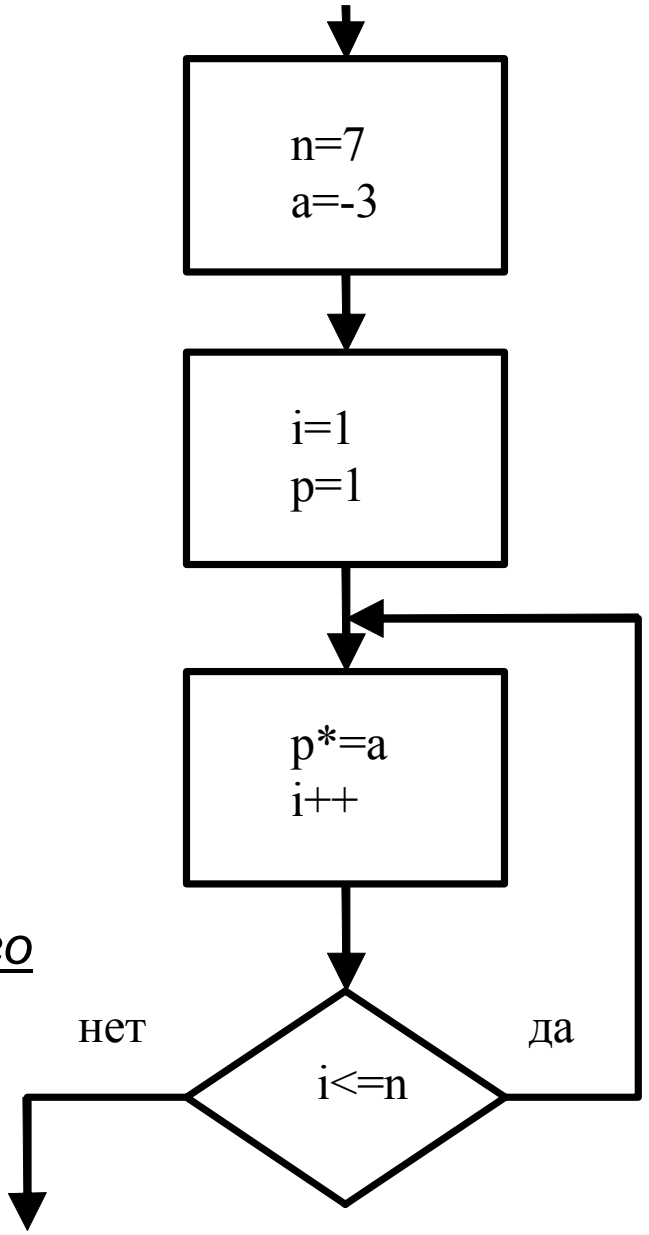
do *оператор;*
while *(условие);*

Пример 1: Вычислить AN:

```
...
n=7;a=-3;
i=1; p=1;
do {
    p*=a;
    i++;
    } while (i<=n);
...
```

Пример 2 (ввод целого положительного числа не больше 4):

```
do
scanf("%d",&i);
while ((i>4)||(i<0)) ;
```



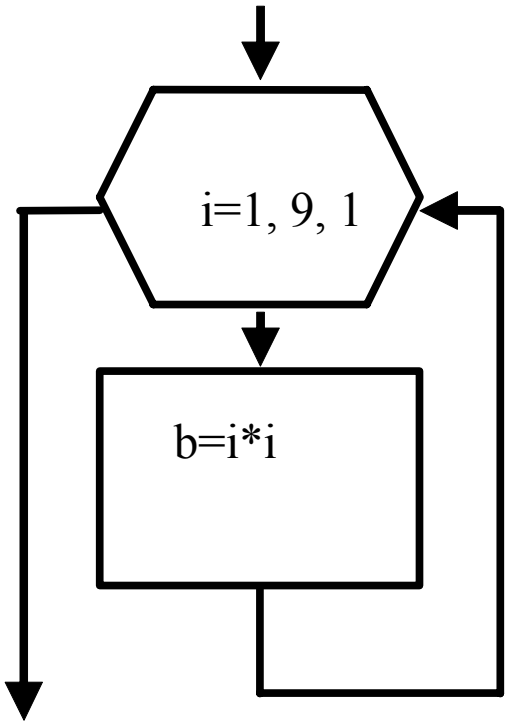
for (выражение1; выражение2; выражение3) оператор;

Где:

Выражение1 – начальные значения;

Выражение2 – условие продолжения цикла

Выражение3 – изменение переменных
в каждой итерации



Пример1: вывод квадратов чисел от 1 до 9

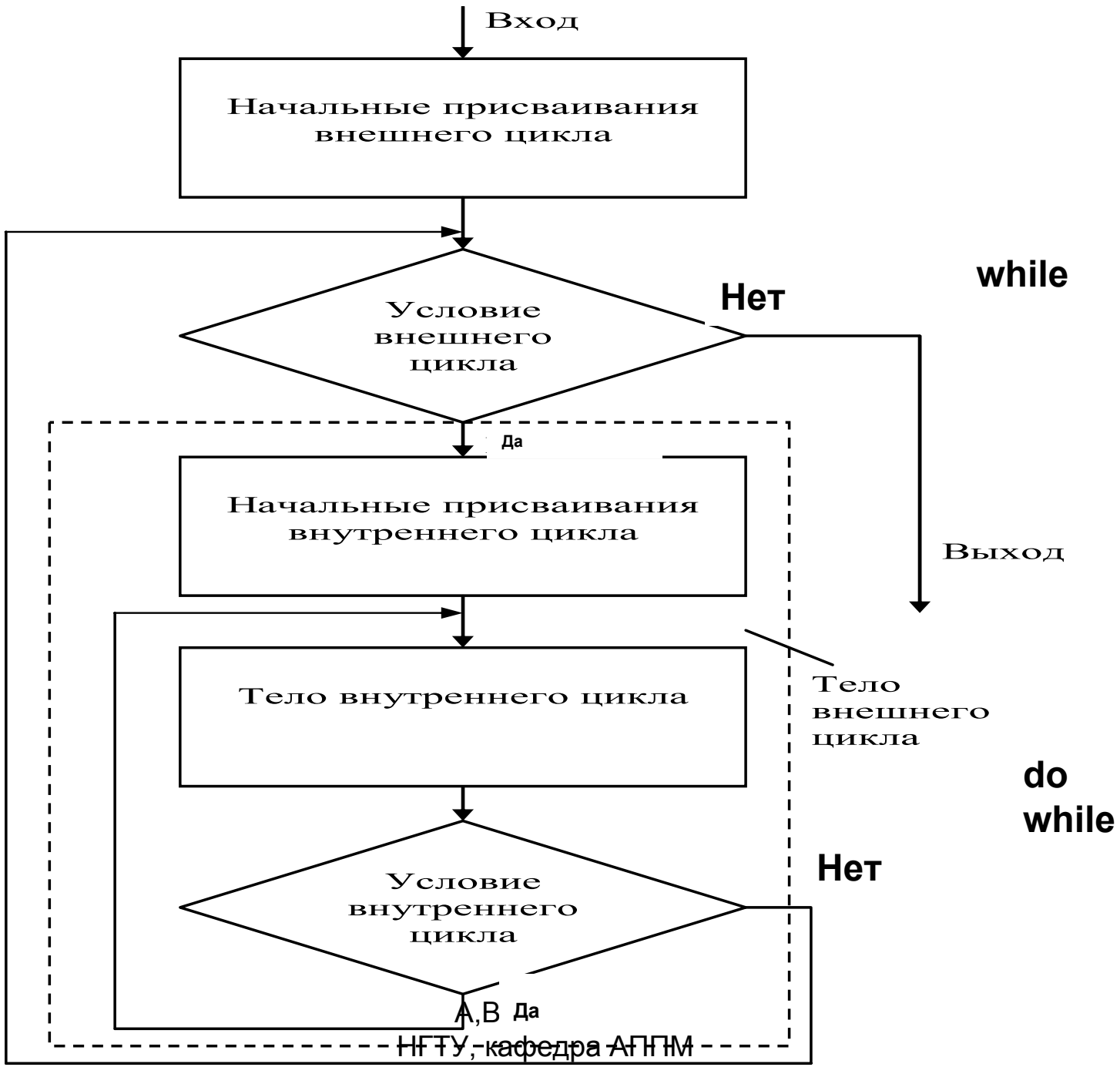
```
void main(void)
    { int i,b;
      for (i=1; i<10; i++)
        {b=i*i;printf("%d ",b);}
    }
```

Пример 2:

```
for (char ch='0'; ch!='a';) scanf("%c", &ch);  // ждет ввода 'a'
```

Пример 3:

```
for (int i=9; i>=0; i--) printf("%d\n", i); // вывод цифр от 9 до 0
```


Вход
Начальные присваивания внешнего цикла
Условие внешнего цикла
Нет
while
Да
Начальные присваивания внутреннего цикла
Выход
Тело внутреннего цикла
Тело внешнего цикла
do while
Нет
Условие внутреннего цикла
A,B Да

Некоторые приемы алгоритмизации

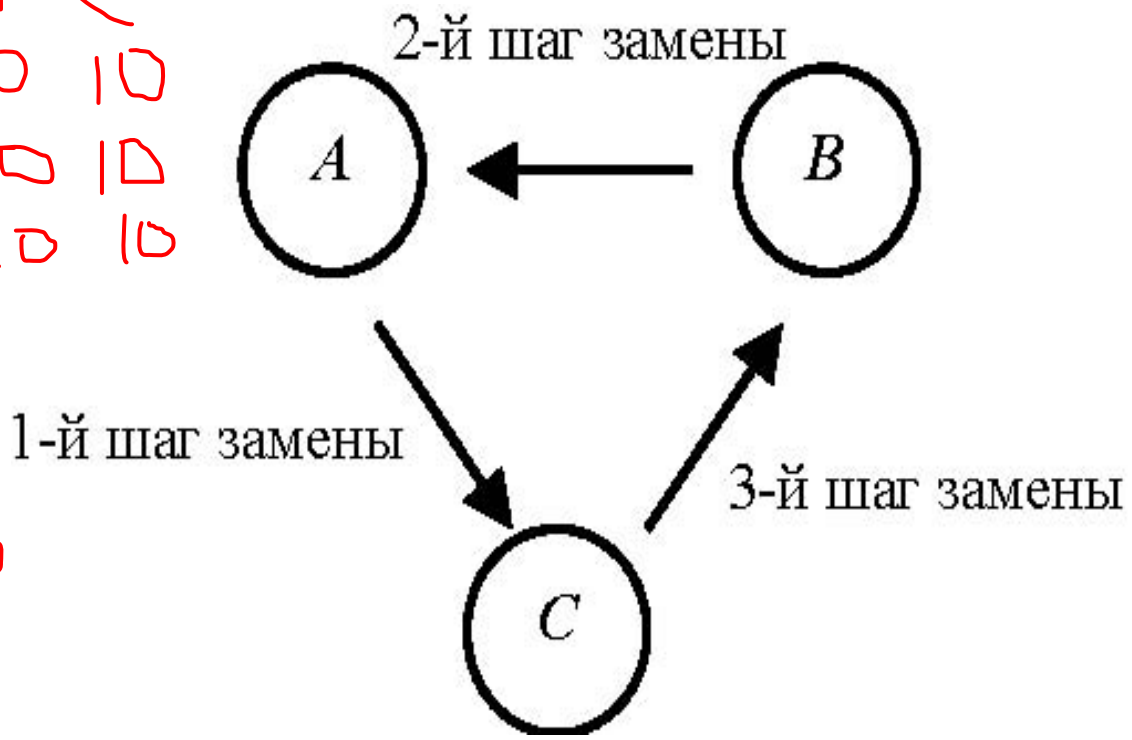
- ***Обмен значениями между двумя переменными A и B***

A B C
10 20 10
20 20 10
20 10 10

- *C = A*
- *A = B*
- *B = C*

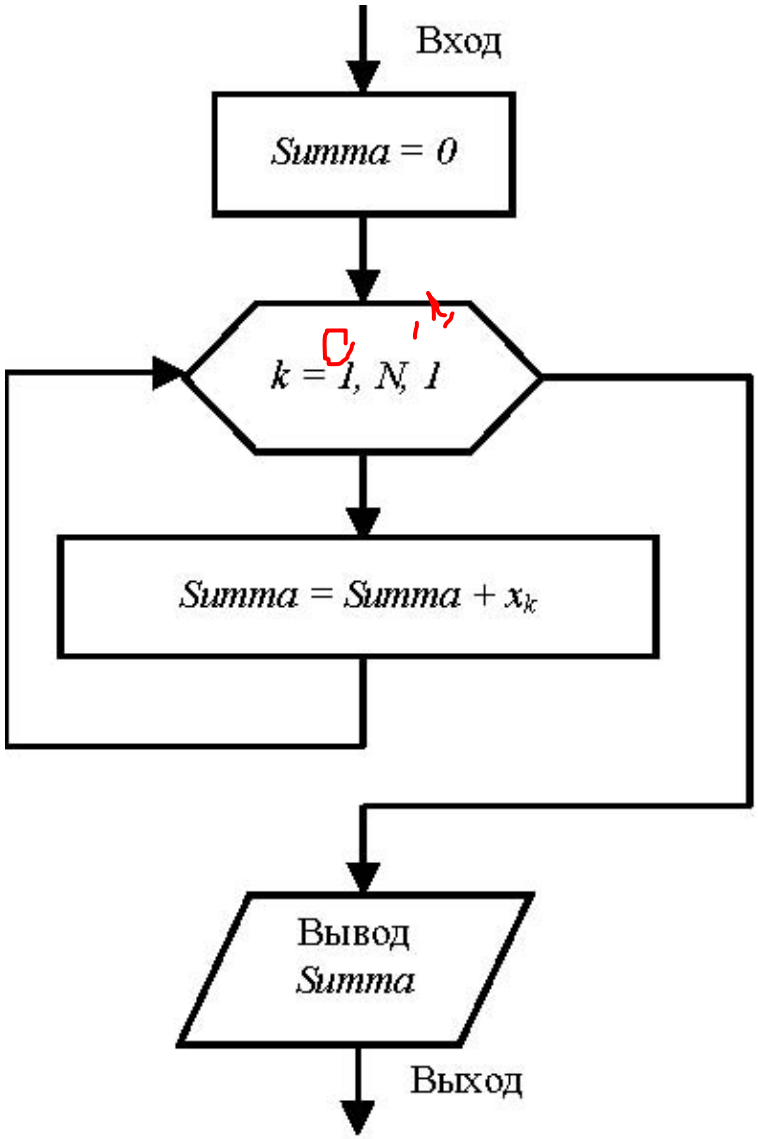
A B
30 20
30 10
20 10

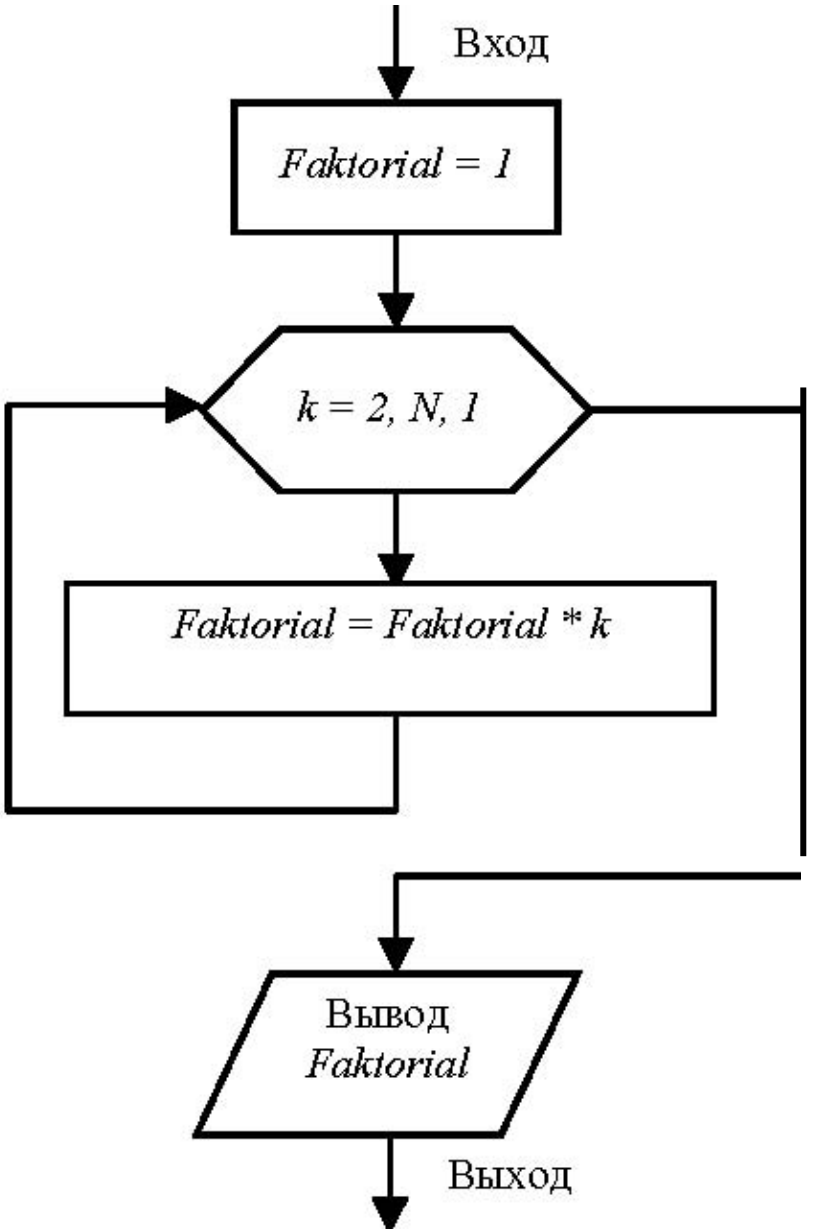
- A=A+B
- B=A-B
- A=A-B



Вычисление произведения (суммы) чисел

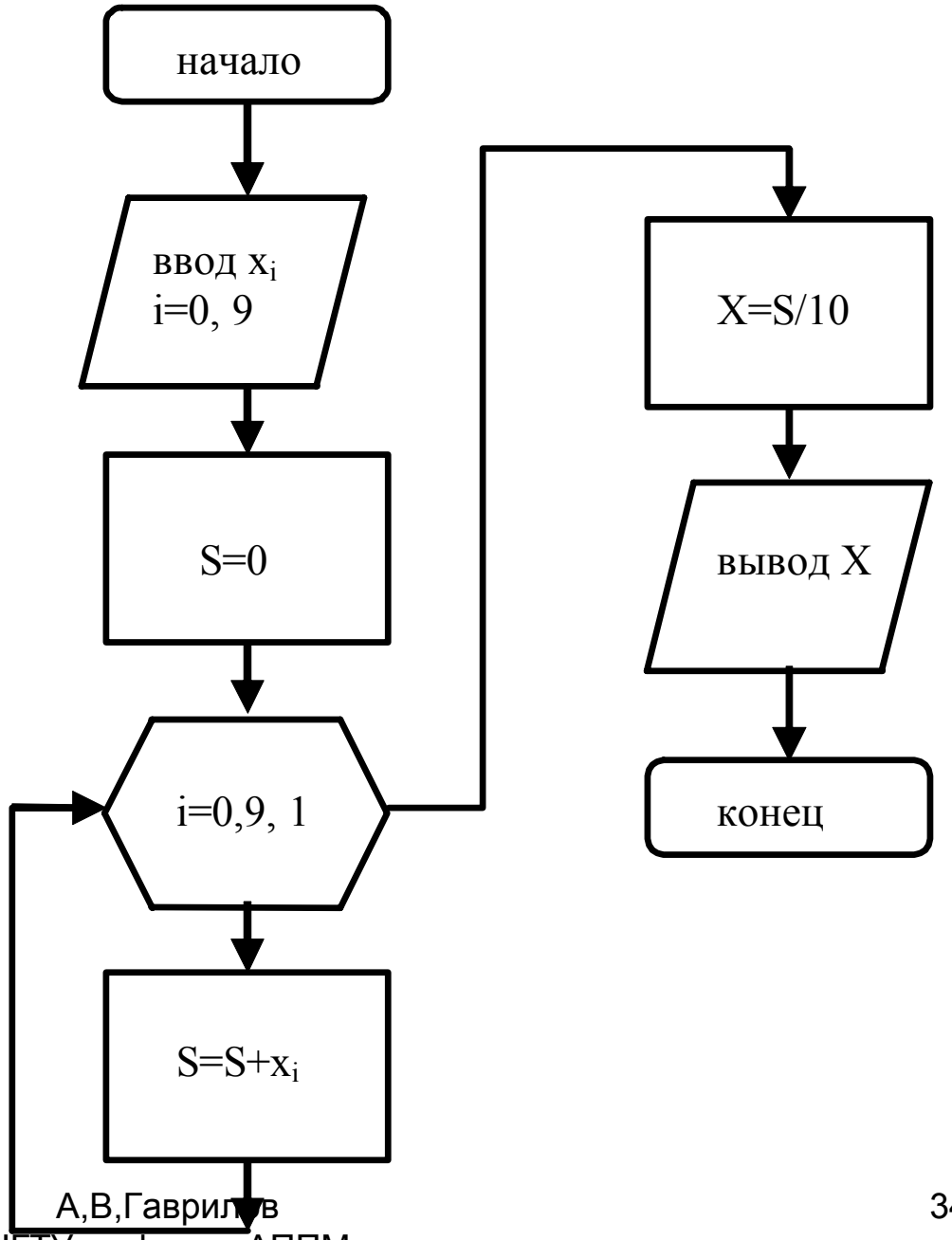
(основаны на накоплении результата в переменной)





$$X = \frac{x_1 + x_2 + \ldots + x_{10}}{10} =$$

$$= \frac{1}{10} \sum_{i=1}^{10} x_i$$



Пример алгоритма нахождения максимального числа в массиве на паскале-подобном псевдокоде

```
// Нахождение максимального элемента списка.
MaxValue := 0;
for j := 1 to N do
    if (Value[j]>MaxValue) then
    begin
        MaxValue := Value[J];
        MaxJ := J;
    end;
```

Рекурсия

- Рекурсия — способ общего определения объекта или действия через себя, с использованием ранее заданных частных определений. Рекурсия используется, когда можно выделить самоподобие задачи

Пример рекурсивного алгоритма

```
procedure CountDown(N : Integer);
begin
   if (N<=0) then exit;
   CountDown(N-1);
end;
```

Эта процедура ничего не делает
Только вызывает сама себя заданное количество раз (N)
При вызове

CountDown(10);

Вызывает себя 10 раз