АЛГОхитрости

Типовые алгоритмические приёмы

При разработке алгоритмов и программ можно выделить **типовые алгоритмические приёмы** – последовательности операций или действий, которые можно применять при алгоритмизации ситуаций или явлений, обладающих сходными характеристиками.

Типовые приёмы наработаны многолетней практикой решения задач на ЭВМ и доведены до наиболее эффективного и экономичного вида.

Рассмотрим некоторые из типовых алгоритмических приёмов

1. **Использование промежуточной переменной (буфера) при обмене значениями двух переменных.**
2. **Сумматор для накопления результатов обработки переменной.**
3. **Вывод на печать текстов с изменяющимся словом.**
4. **Условный оператор: диапазоны, состав числа, високосный год.**
5. **Циклы: ряды, определение количества и состава цифр целого число.**
6. **Циклы: проверка введённых данных.**
7. **Хитрости ЦИКЛевания.**

АЛГОхитрости

Типовые алгоритмические приёмы

- **Использование промежуточной переменной (буфера) при обмене значениями двух переменных:**

A = 3;
B = 6;

Надо поменять значения переменных местами, т.е. сделать:

A = 6;
B = 3;

Для этого действия вводится буфер – переменная K – и выполняются следующие операции:

K = A;
A = B;
B = K;

- **Сумматор – используется для накопления значений в какой либо изменяющейся в процессе обработки переменной, при этом вид обработки может быть самый разный (суммирование, умножение и пр.):**

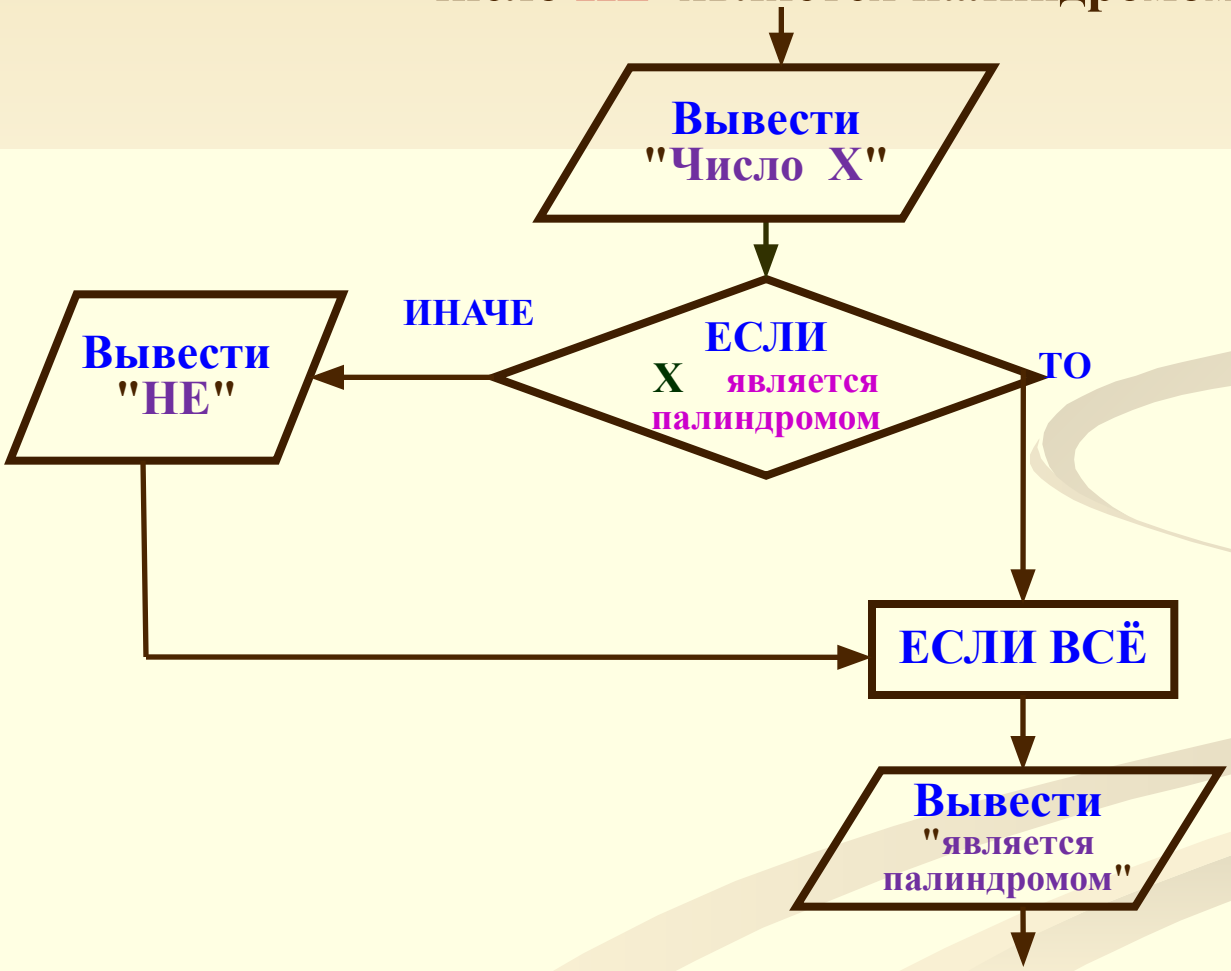
S = S + n;
D = D * h;

АЛГОхитрости

Типовые алгоритмические приёмы

Условный оператор

- **Вывод на печать текстов с одним изменяющимся словом:**
  Пример: - число является палиндромом,
  - число НЕ является палиндромом.



Понятно, что в принципе не очень сложно распространить это приём на вывод в основном одинаковых текстов с несколькими изменяющимися частями

АЛГОхитрости

Типовые алгоритмические приёмы

Условный оператор

- **Ввести исходные данные, найти решение, вывести результат на экран**

1. Определить принадлежит ли x отрезку [-3,7] `(x >= -3) && (x <= 7)`

2. Определить является ли x трёхзначным числом `(x > 99) && (x <= 999)`

3. Дано двузначное число. Определить кратно ли оно трём

Использовать операции if и %

4. Дано двузначное число x. Определить состоит ли оно из одинаковых цифр

```
if (x >= 10) && (x <= 99)
  if x % 10 =  (int) x / 10
    printf ("Да")
  else printf ("Нет");
else printf (" ошибка ввода числа");
```

вариант 2-го if: `if x % 11 = 0`

5. Дано четырёхзначное целое число x. Написать программы на С, определяющие, является ли високосным год с таким номером.

Год является високосным, если его номер кратен 4, однако из кратных 100 високосными являются лишь кратные 400, например. 1700, 1800 и 1900 – не високосные года, 2000 – високосный.

C / C++

АЛГОхитрости

Чтение данных с клавиатуры

Для функции scanf задаются спецификаторы ввода-вывода, которые определяют тип данных вводимых переменных.

При чтении чисел из буфера клавиатуры функция scanf() прекращает чтение числа тогда, когда встречается первый нечисловой символ.

При чтении одиночного символа символы разделителей читаются так же, как и любой другой символ, хотя при чтении данных других типов разделители интерпретируются как разделители полей.

Например, при вводе с входного потока

"x y" фрагмент кода

```
scanf("%c%c%c", &a, &b, &c);
```

помещает символ x в a, пробел в b, а символ y - в c.

Введите конструкцию: A + B разными способами (одним и несколькими операторами ввода С; как в образце; сначала знак – потом цифры и т.д.).

Исследуйте ситуацию, когда знак операции пропадает, а когда – нет!

Напишите ввод цифровых и символьных данных очищая буфер клавиатуры перед вводом знака операции.

Очистка буфера клавиатуры

пока буфер не пуст читает из буфера клавиатуры в оперативную память	`do { q=getchar(); }` `while ( kbhit() )`
Что такое kbhit(), getch()	Требует: `#include <conio.h>`

АЛГОхитрости

Типовые алгоритмические приёмы

Цикл for

❖ **Нарисовать алгоритм, написать программы на C**

1. **Дано целое число N (> 3). Последовательность целых чисел A_k определяется следующим образом:**

$$A_1 = 1,\ A_2 = 2,\ A_3 = 3,\ \dots,\ A_k = A_{k-1} + A_{k-2} - 2*A_{k-3},\quad k = 4, 5, \dots$$

Вывести элементы $A_1,\ A_2,\ \dots,\ A_N$.

2. **Дано целое число N (> 0). Найти и вывести сумму:**

$$1^N + 2^{N-1} + \dots + N^1.$$

Чтобы избежать целочисленного переполнения, вычислять слагаемые этой суммы с помощью вещественной переменной и выводить результат как вещественное число (или использовать длинные целые –long в C).

АЛГОхитрости

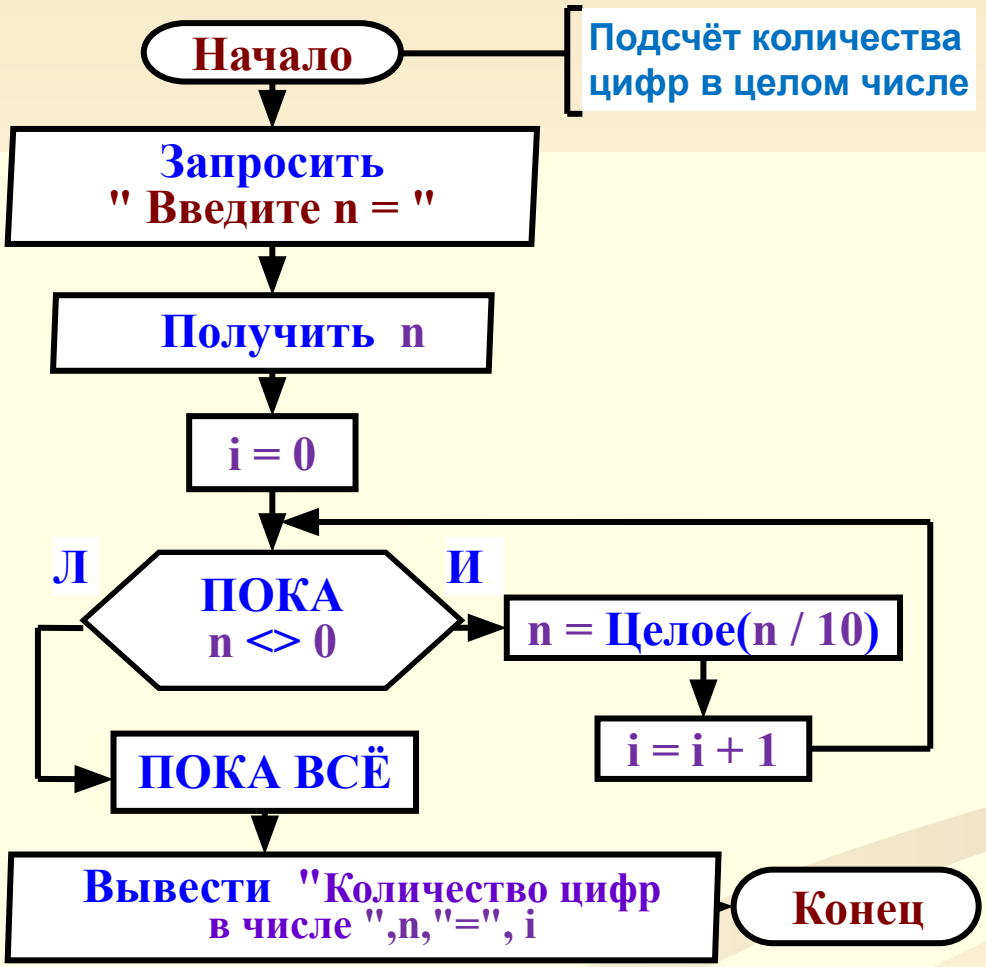
Типовые алгоритмические приёмы. Цикл while

❖ Определение количества цифр целого числа

Введём новые математические операции в псевдокоде:

- **Целое(<выражение>) – в результат заносится только целая часть выражения в круглых скобках.**

Блок-схема



Код

```
// Определить состоит ли число из одинаковых цифр
#include <stdio.h>
main()
{
int   n, i=0, k;
 printf("\nВведите целое число (до 32 767) n=");
 scanf("%D", &n);
   k=n;
     while(k!=0)
     {
       k = (int) k / 10;
       i++;
     }
 printf("\nКоличество цифр в числе %i  равно
                                    %d",n,i);
 return 0;
}
```

АЛГОхитрости

Типовые алгоритмические приёмы. Цикл while

❖ **Выделение цифр, входящих в целое число**

Для решения этой задачи надо:

1. **Получить остаток от деления заданного числа на 10 – это будет правая (последняя) цифра числа (остаток от деления 123 на 10 = 3)**
2. **Убрать из числа последнюю цифру,**
   для этого надо разделить число на 10 и взять целую часть результата деления (123 делить на 10 = 12,3 целая часть нового числа = 12)
3. **Получить остаток от деления нового числа на 10 – это будет правая цифра нового числа (остаток от деления 12 на 10 = 2)**
4. **Убрать из числа последнюю цифру, для чего снова разделить новое число на 10 и взять целую часть результата деления (12 делить на 10 = 1,2 - целая часть нового числа = 1)**
5. **Получить остаток от деления нового числа на 10 – это будет правая цифра числа (остаток от деления 1 на 10 = 1 – *Пояснение*: результат деления = 0, а остаток = 1)**
   (последнее новое число в примере состоит из одной цифры, но без специальных операций это заранее неизвестно, лучше выполнять предыдущие операции)
6. **Теперь надо уяснить как организовать последовательность описанных выше операций: НАДО организовать цикл, в котором:**
   a) **повторять тело цикла до тех пор пока целая часть результата деления нового числа на 10 не станет равна 0,**
   b) **получить остаток от деления текущего числа на 10,**
   c) **получить целую часть от деления текущего числа на 10,**
   d) **назначить целую часть результата деления текущим новым числом,**
   e) **перейти к пункту a).**

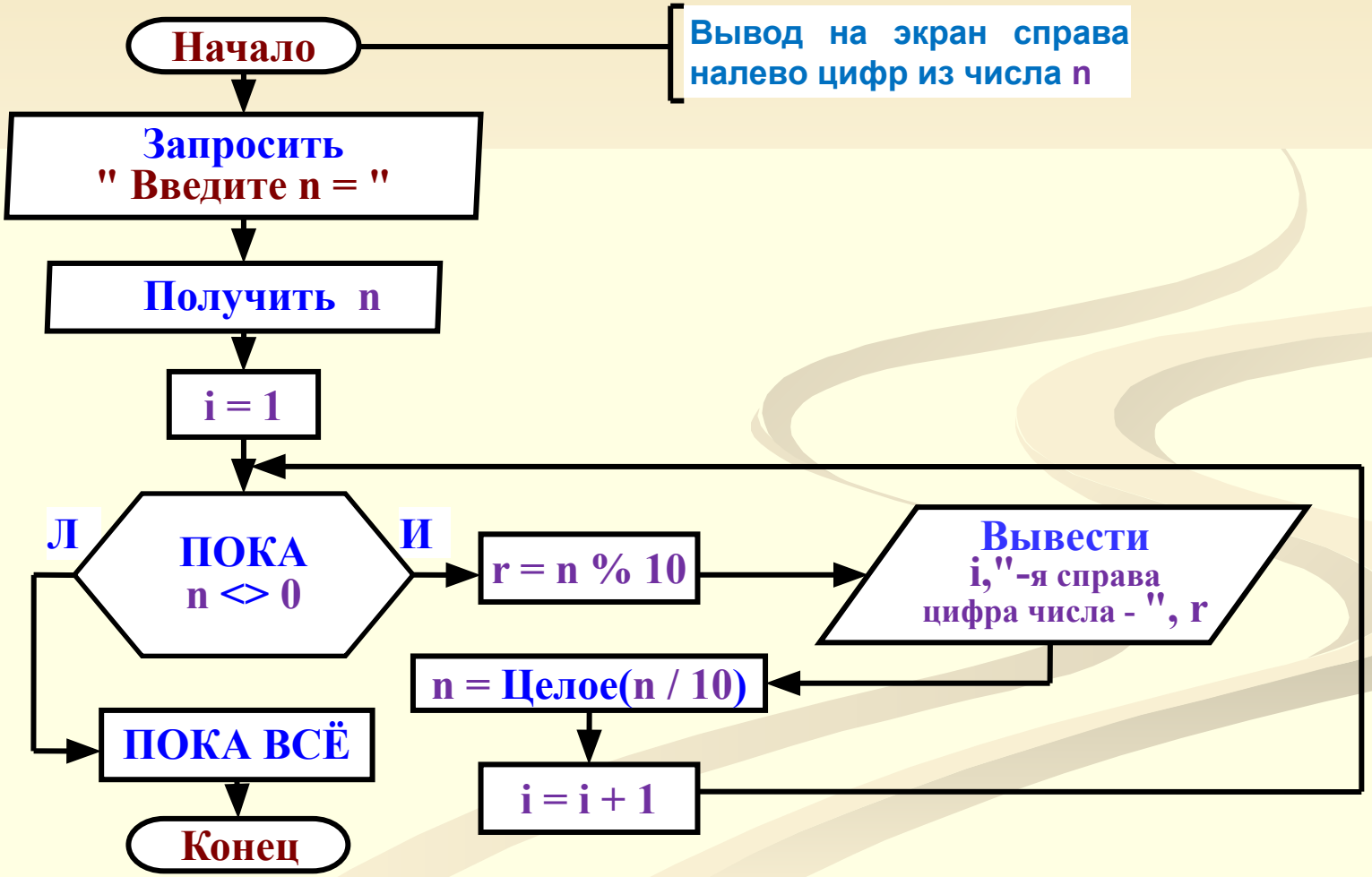
АЛГОхитрости

Типовые алгоритмические приёмы. Цикл while

- **Выделение цифр, входящих в целое число – блок-схема**

Введём новые математические операции в псевдокоде:

- **% или Остаток от деления – в результат записывается остаток от деления операнда_1 на операнд_2 .** Пример: A=243, B=10; r = A % B = 243 % 10 = 3



АЛГОхитрости

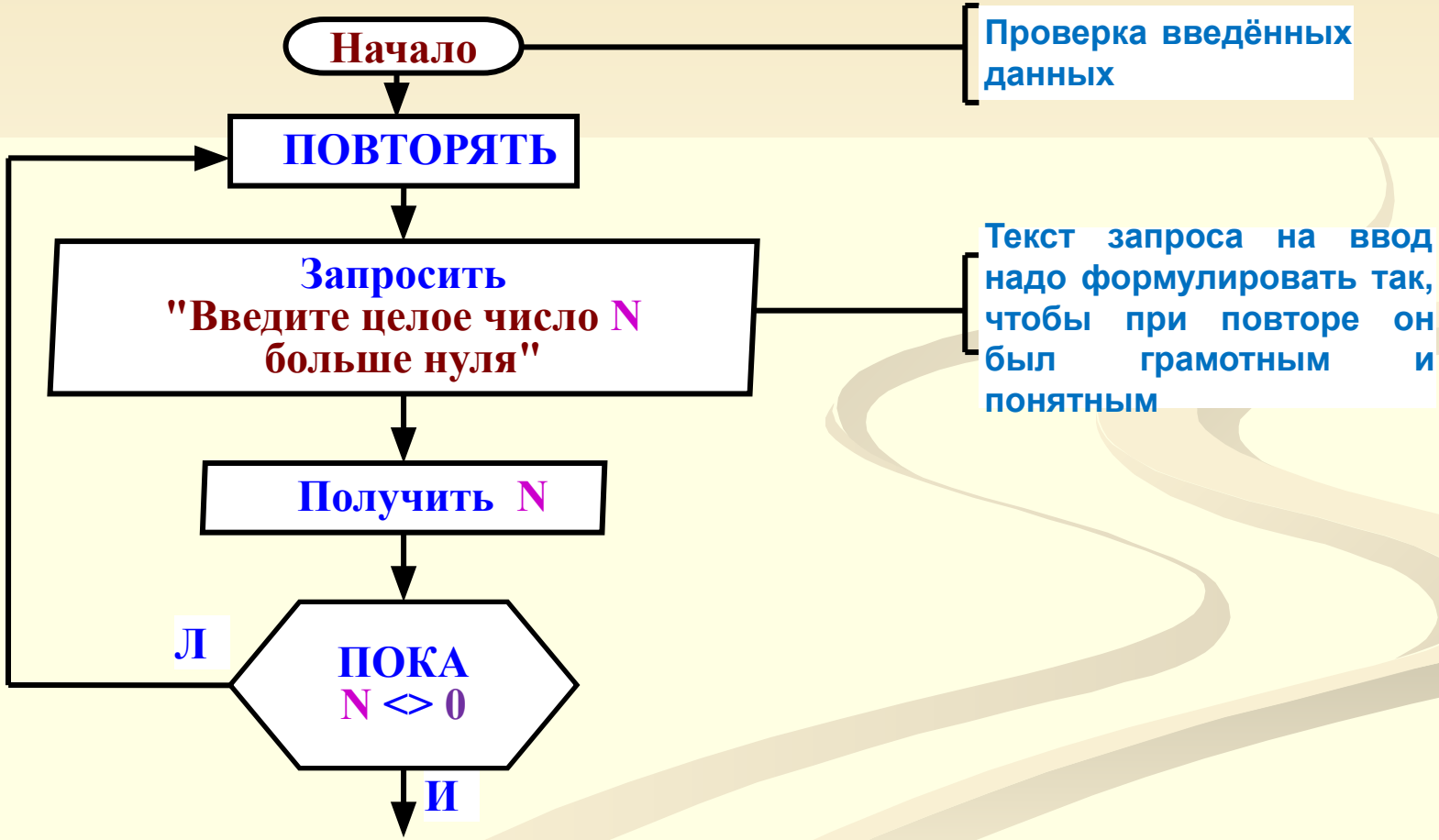
Типовые алгоритмические приёмы

Цикл do-while

- Проверка введённых данных – блок-схема

Это действие осуществляется с использование структуры

Цикл с постусловием



АЛГОхитрости

Хитрости ЦИКЛевания

- Объяснить условие завершения цикла:

```
base = 15;
for  (; base; base--)
        {тело цикла};
```

Цикл будет выполняться пока значение переменной base не станет равно 0, что в С означает false

- Объяснить как можно завершить такой цикл:

```
while (1)
    ……….
    if ( ?????)
            break;
        ……….. ;
```

Вариант:

```
for (;;)
    ……….
    if ( ?????)
            break;
        ……….. ;
```

Цикл может прерван только принудительно, с помощью операторов break или return, используемых при проверке (if, switch/case) условия завершения внутри цикла. Такой цикл может использоваться, когда нет возможности написать одно чёткое условие завершения цикла (например, надо продолжать цикл только в 5-и из 13 веток вложенного оператора switch), а при попадании в остальные ветки – завершать цикл.