ЗАДАЧА 1. Проверить является ли заданное число N палиндромом[1]. Например, числа 404, 1221 — палиндромы.
Исходные данные: N – целое число.

Результаты работы программы: сообщение.

Промежуточные данные: i – параметр цикла, M – переменная для временного хранения значения N, kol – количество разрядов в заданном числе, R= 10^{kol} – старший разряд заданного числа, S – целое число, полученное из цифр числа N , записанных в обратном порядке.

Можно предложить следующий алгоритм решения задачи. Записать цифры заданного числа N в обратном порядке, получится новое число S. Сравнить полученное число S с исходным N. Если числа равны, то заданное число является палиндромом.

Текст программы на языке С:

```
   #include <stdio.h>
   int main()
   {unsigned long int N,M,R,S;
   int kol, i;
   printf("N="); scanf("%ld",&N);
//В цикле вычисляет количество разрядов в числе
//(переменная kol), параллельно с этим
// накапливается вес
// старшего разряда (переменная R).
for (R=1,kol=1,M=N;M/10>0; kol++,R*=10,M/=10);
//В цикле формируется переменная S -  число в
// обратном порядке.
for(S=0,M=N,i=1;i<=kol;S+=M%10*R,M/=10,R/=10,i++);
   //Сравнение исходного числа с перевёрнутым.
   if (N==S) printf("Число - палинром\n");
   else printf("Число не является палиндромом\n");
```

1 Палиндром — это число, слово или фраза одинаково читающееся в обоих направлениях, или, другими словами, любой симметричный относительно своей середины набор символов.

```
return 0;
}
```

Предлагается ещё один алгоритм проверки является ли число палиндромом. Сравниваем левую и правую цифры числа, если встречаем несовпадение – исходное число не является палиндромом.

Код программы с минимальными комментариями приведён ниже. Читателю предлагается самостоятельно изучить программу.

```
#include <stdio.h>
int main()
{unsigned long int N,M,R;
int kol, i, pr, lev, prav;
printf("N="); scanf("%ld",&N);
for (R=1,kol=1,M=N;M/10>0; kol++,R*=10,M/=10);
//Переменная pr отвечает за совпадение левой и
правой цифры числа
for(M=N,pr=i=1;i<=kol;i+=2)
{
//В переменной prav хранится текущая правая
// цифра числа.
   prav=M%10;
//В переменной lev хранится текущая правая
// цифра числа.
   lev=M/R;
   if (lev!=prav) {pr=0;break;}
//Отрезаем левую цифру числа
   M-=(M/R)*R;
//Отрезаем правую цифру числа
   M/=10;
//Вес левого разряда уменьшаем в 100 раз
   R/=100;
}
if (pr) printf("Число - палинром\n");
else printf("Число не является палиндромом\n");
return 0;
}
```

ЗАДАЧА 2. Вводится последовательность целых чисел, 0 – конец последовательности. Найти наименьшее число среди положительных, если таких значений несколько[2], определить, сколько их.
Исходные данные: `N` – текущий текущий элемент последовательности.

Результаты работы программы: `Min` – минимальный положительный элемент последовательности, `K` – количество значенийравных минимуму.

Блок-схема решения задачи приведена на рис. 1.

2 Предположим вводится последовательность чисел 11, -3, 5, 12, -7, 5, 8,-9, 7, -6, 10, 5,0. Наименьшим положительным числом является 5. Таких минимумов в последовательности 3.

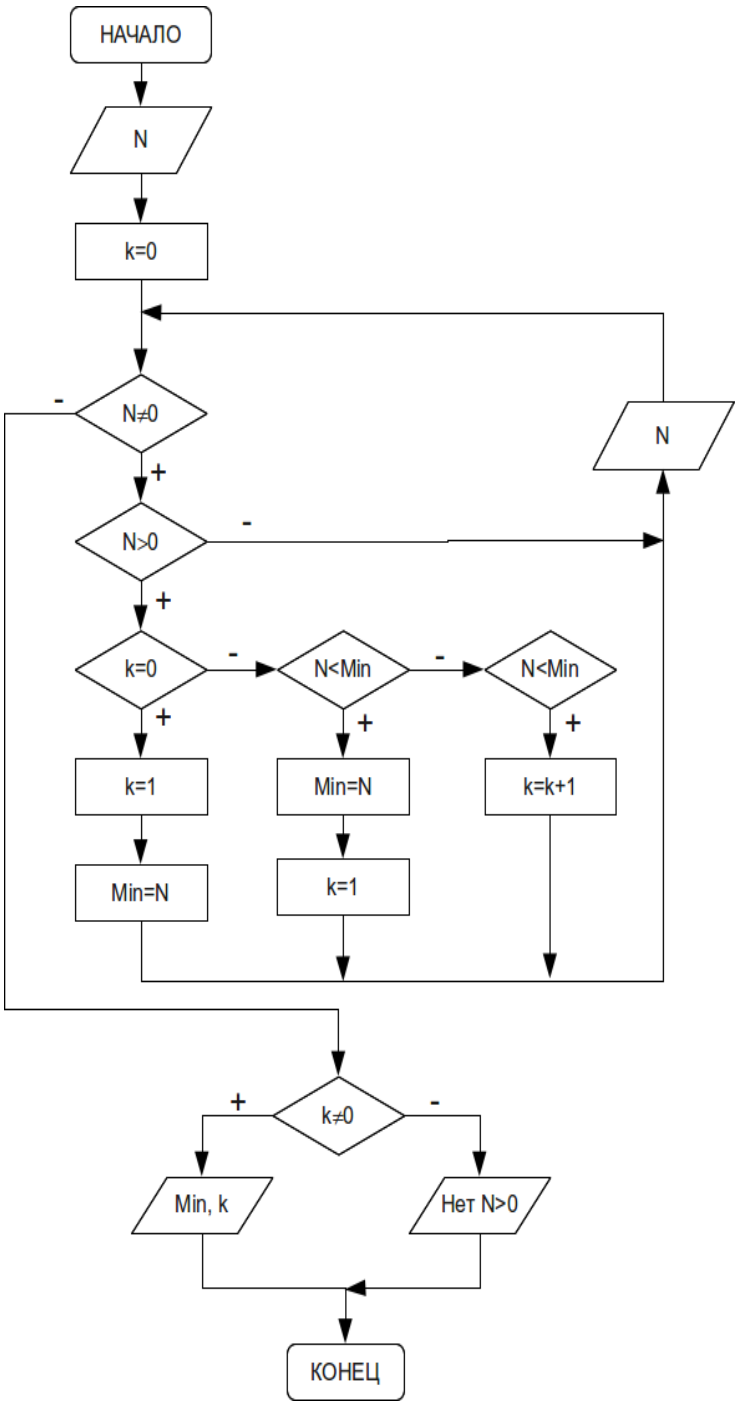

Рис. 1 Алгоритм поиска минимального положительного числа в последовательности

Далее приведен текст подпрограммы .

```
#include <stdio.h>
int main()
{
   int N,Min,K;
//Предположим, что в последовательности нет
// положительных чисел, K=0. Переменная K
// содержит количество значений
```

```
//равных минимуму. Вводим число N и если оно не
// равно нулю
printf("N=");scanf("%d",&N);
for (K=0;N!=0;)
{
//проверяем является ли оно положительным.
      if (N>0){
//если K=0, поступил 1-й положительный элемент,
// который и объявляем минимальным.
       if (K==0) {K=1;Min=N;}
//если текущее положительное число элемент не
//первое сравниваем его с предполагаемым
// минимумом,если текущее число меньше,
//записываем его в Min
//и сбрасываем счетчик K в единицу
       else if (N<Min) {Min=N;K=1;}
//если положительный элемент равен минимуму,
// увеличиваем значение счетчика K.
       else if (N==Min) K++;
      }
      printf("N=");scanf("%d",&N);
}
//Конец цикла
//Если значение счетчика не равно нулю,
// выводим значение минимального элемента
//и количество таких элементов.
   if (K!=0) printf("Min=%d\tK=%d\n",Min,K);
//в противном случае сообщаем об отсутствии
// положительных элеменнтов.
else printf("Нет положительных элементов \n");
return 0;
}
```

ЗАДАЧА 3. Поступает последовательность из N вещественных чисел. Определить является ли она возрастающей.
Исходные данные: `N` (целое число) – количество элементов последовательности; `B` (вещественное число) определяет текущий элемент последовательности.

Результаты работы программы: `Pr` (целое число) определяет является ли последовательность возрастающей.

Промежуточные данные: `i` – параметр цикла, номер вводимого элемента последовательности, `A` – значение предыдущего элемента последовательности.

В возрастающей последовательности каждое число больше предыдущего. Если встретится число, которое окажется меньше или равно предыдущего, то последовательность окажется невозрастающей.

Алгоритм решения задачи следующий.

Вводится N – количество чисел в последовательности. Предполагаем, что последовательность возрастающая (Pr=1). Далее организуется цикл, который повторяется N раз (переменная цикла i меняется от 1 до N с шагом 1). На каждой итерации выполняются следующие действия.

1. В переменную B вводится очередной элемент последовательности.
2. На всех итераций (кроме первой) сравниваем текущее значение последовательности (B) с предыдущим (A). Если $B \le A$, то наша последовательность невозрастающая (в переменную Pr записываем 0).
3. В переменную A (предыдущее значение элемента последовательности) сохраняем значение B (текущее значение элемента последовательности).

После выхода из цикла анализируем значение Pr. Если Pr=1, последовательность возрастающая; иначе последовательность не является возрастающей.

Текст программы с комментариями приведён ниже.

```
   #include <stdio.h>
   int main()
   {
      int i,N,Pr;
      float B,A;
//Ввод N-количества элементов последовательности.
      printf("N=");scanf("%d",&N);
   //Цикл, который повторяется N раз.
//Предполагаем, что последовательность
// возрастающая
      for(Pr=i=1;i<=N;i++)
      {
   //Ввод очередного элемента последовательности
         printf("B=");scanf("%f",&B);
   //Начиная со второго элемента
         if (i!=1)
//Сравниваем текущий элемент (B) с предыдущим (A).
//Если текущий элемент меньше или равен
//предыдущего, то последовательность не является
// возрастающей (Pr=0).
         if (B<=A) Pr=0;
//В переменную A сохраняем текущее значение
//элемента
// последовательности (B).
         A=B;
      }
//Проверка является ли последовательность
// возрастающей
// (по значению переменной Pr)
if (Pr)
printf("Последовательность возрастающая\n");
      else
printf
("Последовательность не является возрастающей\n");
```

```
    return 0;
}
```

ЗАДАЧА 4. Задано число N в десятеричной системе. Перевести число в восьмеричную систему счисления.
Исходные данные: N – исходное целое число.

Результаты работы программы: S – число в восьмеричной системе.

Промежуточные данные: T – очередная цифра числа, R – вес цифры числа.

Алгоритм перевода следующий в восьмеричную систему следующий.

1. Определить очередную цифру числа (T) в восьмеричной системе счисления, вычислив остаток от деления числа N на 8.
2. Уменьшить число N в 8 раз.
3. Если N>0, то повторяем пункты 1-3. Иначе мы определили все разряды числа.

Чтобы реализовать в виде программы алгоритм, представленный пунктами 1-3, необходимо собрать число из получаемых в п.1 цифр. Для этого надо просуммировать все цифры числа, умноженные на их вес. Вес первой (младшей) числа равен 1 (10^0), второй – 10 (10^1), третьей – 100 (10^2) и т д.

На рис. 2 Приведен пример перевода числа 256 заданного в десятичной системе счисления в восьмеричную. В результате получим, $256_{(10)}=400_{(8)}$.

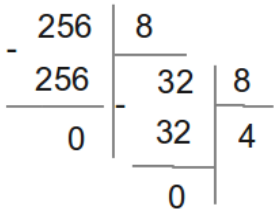

Рис. 2 Пример перевода числа в новую систему счисления

Формальный алгоритм можно записать так.

1. Ввод числа `N`. Копируем значение `N` в переменную `M`. Определяем вес младшего разряда `R=1`. Число в восьмеричной системе системе будет представлять из себя сумму цифр умноженных на их вес (степени числа 10). Поэтому в переменную `S` запишем число `0`.
2. Пока M>0, повторяем пункты 3-4.
3. Определяем очередную цифру числа `T=M%8`. Накапливаем число S в восьмеричной системе счисления S+=T*R. Увеличиваем вес следующего разряда числа `R*=10`.
4. Уменьшить число M в 8 раз.
5. Вывод S. В S хранится число в восьмеричной системе счисления.

Рассмотрим текст программы с комментариями.

```
   #include <stdio.h>
   int main(int argc, char **argv)
   {
      long int N,S,R,M;
      int T;
//Ввод исходного числа N.
      printf("N=");scanf("%ld",&N);
//Сохраняем копию исходного числа в переменной M.
      M=N;
//В переменной R храним вес очередного разряда
// числа.
//Первоначальное значение R равно 1.
//S=0, в переменной S будет собираться число в
// восьмеричной системе счисления.
      for(R=1,S=0;M>0;M/=8)
      {
//Определяем T – очередную цифру числа.
         T=M%8;
//К переменной S добавляем цифру, умноженную на
// вес.
         S+=T*R;
   //Вес следующей цифры увеличиваем в 10 раз.
```

```
      R*=10;
   }
//Вывод числа в восьмеричной системе счисления.
   printf("S=%ld\n",S);
   return 0;
}
```

Программа и алгоритм мало изменятся при переводе из десятеричной в р-чную $2 \le p \le 9$ систему счисления. Необходимо в коде просто заменить цифру 8 заменить на основание другой системы счисления. Текст программы перевода числа из десятеричной в р-чную систему счисления приведён ниже без комментариев.

```
#include <stdio.h>
int main(int argc, char **argv)
{
   long int N,S,R,M;
   int T,p;
   printf("N=");scanf("%ld",&N);
   printf("p=");scanf("%d",&p);
   M=N;
   for(R=1,S=0;M>0;M/=p)
   {
      T=M%p;
      S+=T*R;
      R*=10;
   }
   printf("S=%ld\n",S);
   return 0;
}
```

ЗАДАЧА 5. Задано число N в восьмеричной системе счисления. Перевести число в десятеричную систему счисления.
Исходные данные: `N` – целое в восьмеричной системе сисчисления.

Результаты работы программы: `S` – число в десятеричной системе.

Промежуточные данные: T – очередная цифра числа, R – вес цифры числа.

Алгоритм перевода из восьмеричной в десятеричную.

1. Ввод числа `N`. Копируем значение `N` в переменную `M`. Определяем вес младшего разряда `R=1`. Число в десятеричной системе системе будет представлять из себя сумму цифр умноженных на их вес (степени числа 8). В переменную `S` запишем число `0`.
2. Пока M>0, повторяем пункты 3-4.
3. Определяем очередную цифру числа `T=M%10`. **Накапливаем число S в восьмеричной системе счисления** `S+=T*R`. **Увеличиваем вес следующего разряда числа** `R*=8`.
4. Уменьшить число M в 10 раз.
5. Вывод S. В S хранится число в десятеричной системе счисления.

Код программы приведён ниже.

```
#include <stdio.h>
int main(int argc, char **argv)
{
   long int N,S,R,M;
   int T;
//Ввод исходного числа N.
   printf("N=");scanf("%ld",&N);
//Сохраняем копию исходного числа в переменной M.
   M=N;
//В переменной R храним вес очередного разряда
// числа.
//Первоначальное значение R равно 1.
//S=0, в переменной S будет собираться число в
```

```
// десятеричной системе счисления.
      for(R=1,S=0;M>0;M/=10)
      {
//Определяем T – очередную цифру числа.
         T=M%10;
//К переменной S добавляем цифру,
//умноженную на вес.
         S+=T*R;
   //Вес следующей цифры увеличиваем в 8 раз.
         R*=8;
      }
   //Вывод числа в десятеричной системе счисления.
      printf("S=%ld\n",S);
      return 0;
   }
```

Ниже приведён текст программы перевода числа из p-чной ($2 \le p \le 9$) системы счисления в десятеричную без комментариев.

```
   #include <stdio.h>
   int main(int argc, char **argv)
   {
      long int N,S,R,M;
      int T,p;
      printf("N=");scanf("%ld",&N);
      printf("p=");scanf("%d",&p);
      M=N;
      for(R=1,S=0;M>0;M/=10)
      {
         T=M%10;
         S+=T*R;
         R*=p;
      }
      printf("S=%ld\n",S);
      return 0;
   }
```