5. ПОСЛЕДОВАТЕЛЬНАЯ ОБРАБОТКА ДАННЫХ

5.1. Приемы последовательной обработки данных

В ряде задач последовательность элементов исходных данных вводится и обрабатывается по одному элементу в том порядке, в каком она размещена в файле на внешнем носителе, причем каждый элемент используется не более одного раза.

В этом случае в оперативной памяти не требуется хранить сразу все элементы. Достаточно иметь одну переменную, содержащую текущий (очередной) элемент входной последовательности, в которую по очереди вводятся все входные элементы.

В некоторых случаях используются несколько текущих элементов (например, два-три).

Такие задачи называют *(однопроходной) последовательной обработкой данных*. Элементами данных могут быть числа; символы, образующие текст; записи файла и другие величины различных типов.

Последовательность исходных данных может задаваться с указанием количества элементов, с признаком конца или обрабатываться до конца входного файла. Рассмотрим записанные на псевдокоде укрупненные алгоритмы последовательной обработки для этих трех случаев.

1. Пусть входная последовательность задается с указанием количества элементов $n \geq 0$ в следующем порядке:

n, X1, X2, ... , Xn

Алгоритм последовательной обработки удобно записывать в виде цикла с параметром (алг. 4.1). Пример - ввод массива x в программе 4.1.

Алгоритм 1 Последовательная обработка заданного количества элементов

```
Ввод n;
for (j=1; j<=n; j++)
{   Ввод X;
    Обработка X;
}
```

2. Входная последовательность задается с признаком конца:

X1, X2, ... , Xn, W

где n - неизвестное заранее количество элементов ($n \geq 0$), W - признак конца последовательности (известное заранее особое значение элемента, отличающееся от элементов последовательности).

Алгоритм 2. Последовательная обработка элементов с признаком конца W

Ввод X;
while (X != W) {
  Обработка X;
  Ввод X;
}

Иногда в языке С можно обойтись одной операцией ввода X, поместив ее внутри условия цикла. При n ≥ 1 возможен цикл с постусловием.

3. Входная последовательность X1, X2, ... , Xn продолжается до конца файла, n - неизвестное заранее количество элементов (n ≥ 0), признак конца отсутствует.

При вводе данных с клавиатуры конец файла задается комбинацией клавиш Ctrl-Z (или Ctrl-z), затем нажимается клавиша Enter.

В языке С конец входного файла обнаруживается обычно только после попытки ввода данных за пределами файла (алг. 4.3). Например, значением стандартной функции ввода scanf является количество фактически введенных элементов, которое можно проверить после ввода. Если не удалось ввести ни одного элемента, значение scanf равно EOF, обычно –1 (пример 4.1).

Алгоритм 3. Последовательная обработка элементов до конца файла
Ввод X;
while (не конец входного файла)
{ Обработка X;
   Ввод X;
}

Иногда можно обойтись одной операцией ввода X внутри условия цикла, совместив ввод с проверкой конца файла, как в программе 4.1. Другие примеры приведены в программах.

Рассмотрим последовательную обработку на примере числовых данных. Входные числа разделяются любым количеством пробелов, символов табуляции и новой строки, и поэтому их можно вводить с помощью простого вызова функции scanf. Другие правила ввода требуют более сложного программирования (см. программу 4.5).

Примеры последовательной обработки символьной информации приведены в программах 4.3, 4.5, 4.6.

5.2. Примеры последовательной обработки данных

Решение задачи полезно начинать с составления тестов. Это облегчает понимание задачи и разработку программы. Для изучения программы также желательно тестировать ее вручную.

Пример 1. Последовательность вещественных чисел продолжается до конца файла. Составить программу нахождения максимального члена последовательности.

Тест. Вход: -5 3.1 2
Выход: Максимум= 3.100000

В программе 4.1 использован следующий метод определения максимума. Пусть max обозначает максимальный член просмотренной части последовательности. Если очередное число оказывается больше max, оно присваивается этой переменной. Начальное значение max равно первому члену последовательности.

Программа работает без диалога (в пакетном режиме).

Программа 1

```
/*                  Определение максимального числа          */
#include <stdio.h>

void main(void)
{   float x;          /* Текущее число                        */
    float max;        /* Текущий максимум                     */
    int k;            /* Количество введенных чисел           */

    k = scanf("%f", &x);
    if (k < 1)                                  /* Нет чисел  */
       printf("\nВходная последовательность пуста\n");
    else
    {  max = x;                                 /* 1-е число  */
       while (scanf("%f", &x) > 0)
          if (max < x)   max = x;
       printf ("\nМаксимум= %f\n", max);
    }
 }
```

Пример 2. Последовательность неотрицательных вещественных чисел заканчивается числом -1. Составить программу нахождения ближайшего к началу последовательности локального максимума, т. е. номера и значения ближайшего к началу члена последовательности, значение которого больше обоих его соседей.

Тест 1. Вход: 4 3.2 5 2 8 6 -1
Выход: 1-й локальный максимум= 5.000000, 3-е число

Тест 2. Вход: 2 3.2 -1
Выход: Локальный максимум отсутствует

Очередное число должно сравниваться с двумя соседями. Поэтому в программе 4.2 хранятся *три соседних* текущих числа – типовой прием. Пока не обнаружен максимум (признак e равен 1), среднее из трех чисел x запоминается в качестве кандидата на максимум, и увеличивается номер

числа j, для которого e служит шагом. При обнаружении первого локального максимума признак e получает значение 0, и поэтому найденные значения max и j больше не изменяются до конца входной последовательности.

Программа 2

```
/*          Нахождение 1-го локального максимума чисел          */
#include <stdio.h>
#define  W  -1          /* Признак конца чисел                    */
int main (void)
{   float x, xpr, xsl;   /* Текущее, предыдущее и следующее число  */
    float max;           /* 1-й локальный максимум (значение x)     */
    int j;               /* Номер локального максимума              */
    int k;               /* Количество введенных чисел              */
    int e;               /* 0 - есть локальный максимум, 1 - нет    */

    e = 1;
    k = scanf ("%f  %f  %f", &xpr, &x, &xsl);
    if (k < 3)   xsl = W;                    /* < 2 чисел          */
    for (j=2;   xsl != W;   j=j+e)
    {  if (e==1)   max = x;          /* Запоминание x              */
       if (x > xpr)
          if (x > xsl)               /* Нашли локальный максимум   */
             e = 0;          /* Прекратить запоминание x и его номера */
       xpr = x;   x = xsl;
       scanf ("%f", &xsl);
    }
    if (e == 0)
       printf  ("\n1-й локальный максимум= %f, %d-е число\n", max,  j);
    else
       printf  ("\nЛокальный максимум отсутствует \n");
   return 0;
}
```

Пример 3. Подсчет строк, слов и символов. Составить программу подсчета во входном тексте количества строк, слов и символов. Словом считается любая последовательность символов, не содержащая разделителей: пробелов, символов табуляции и новой строки. Строка заканчивается символом новой строки. Эта простая программа играет важную роль в операционной системе UNIX.

Тест. Вход:

Если друг оказался вдруг
И не друг, и не враг, а так

Выход:

Строк: 2, слов: 12, символов: 53

Различные решения задачи можно получить, используя разные варианты описания структуры (грамматики) входного текста на метаязыке МБНФ (см. разделы 2.2 и 5.2).

При обработке символьной (текстовой) информации структура алгоритма повторяет структуру читаемого текста. Каждому знаку повторения "..." в правилах грамматики, описывающей структуру текста, в алгоритме чтения и обработки текста соответствует цикл, знаку "|" (или) – ветвление.

Данный текст можно рассматривать на одном уровне: как последовательность символов, либо как двухуровневую структуру: слова, состоящие из символов, либо как трехуровневую структуру: строки - слова – символы.

Подсчет строк не вызывает трудностей: их количество равно числу символов новой строки. Труднее подсчитать слова: их количество не равно числу разделителей (разделители могут следовать подряд).

Поэтому данной задаче больше соответствуют одно- или двухуровневая структура текста. В качестве примера используем одноуровневую структуру (программа 4.3).

Количество слов равно числу их начальных символов, т. е. таких ситуаций, когда символ слова следует за разделителем.

Программа 3

```
/*                  Подсчет строчек, слов и символов          */
/*     Грамматика входного текста на метаязыке МБНФ:          */
/*  текст          ::= [символ...] EOF                        */
/*  символ         ::= разделитель | символ-слова             */
/*  разделитель ::= пробел | новая-строка | табуляция         */
/*  символ-слова - любой символ, кроме разделителей           */

#include <stdio.h>
#define  DA   1
#define  NET  0

void main (void)
{   int sim;                    /* Текущий символ (int для EOF)  */
    int kstr, ksl, ksim;        /* Кол-во строк, слов и символов */
    int razdel;                 /* Символ - разделитель          */

    razdel = DA;                /* 1-й символ текста - начальный */
    kstr = ksl = ksim = 0;
    while ((sim = getchar()) != EOF)
    {   ksim++;
        switch (sim)
```

```
        {  case '\n':
               kstr++;
               razdel = DA; break;          /* Эту строчку можно убрать!        */
           case ' ':
           case '\t':
               razdel = DA; break;
           default:                              /* Символ слова                 */
               if (razdel)  /* Предыдущий символ - разделитель                  */
               {  razdel = NET;
                  ksl++;
               }
        }
    }
    printf ("Строк: %d, слов: %d, символов %d \n", kstr, ksl, ksim);
}
```

Упражнения и задачи

5.1. Дана последовательность вещественных чисел с указанием их количества. Составить программу

а) определения среднего арифметического значения чисел;

б) определения произведения чисел, не превышающих 2.

5.2. Дана последовательность ненулевых целых чисел, заканчивающаяся числом 0. Составить программу

а) определения минимального из четных по значению чисел;

б) определения количества чисел, делящихся на 5.

5.3. Дана последовательность целых чисел, занимающая весь файл. Составить программу определения, имеются ли в последовательности два соседних числа, первое из которых на 10 больше второго. Результатом работы программы является слово "ДА" или слово "НЕТ".

5.4. Дано целое n>0 и последовательность из n действительных чисел. Составить программу для определения, образуют ли числа знакочередующуюся последовательность.

5.5. Дано целое N>0 и последовательность из N целых чисел. Составить программы вычисления следующих величин.

а) Сумма чисел, кратных 5.

б) Сумма чисел, являющихся нечетными и отрицательными.

в) Сумма чисел, удовлетворяющих условию: модуль числа меньше его номера.

г) Количество чисел, являющихся отрицательными.

д) Количество чисел, являющихся нечетными.

е) Количество чисел, кратных 3 и не кратных 5.

ж) Произведение чисел, имеющих четные порядковые номера и являющихся нечетными числами.

з) Сумма и количество чисел, кратных 5.

и) Количество положительных и количество отрицательных чисел.

к) Порядковые номера и количество чисел, кратных трем.

5.6. Дано целое N>0 и последовательность из N действительных чисел. Составить программы для получения следующих величин.

а) Среднее арифметическое квадратов положительных чисел.

б) Сумма элементов с четными номерами и суммы элементов с нечетными номерами.

в) Разность наибольшего и наименьшего числа.

г) Наибольшее среди четных по порядку чисел.

д) Наименьшее среди нечетных по порядку чисел.

е) Наибольшее среди модулей (абсолютных величин) чисел.

ж) Наибольшее среди отрицательных чисел.

з) Сумма положительных и количество отрицательных чисел.

5.7. Дана последовательность действительных чисел, продолжающаяся до конца файла. Составить программы для решения следующих задач.

а) Определить порядковый номер наименьшего из них.

б) Определить, образуют ли они возрастающую последовательность.

в) Определить, со скольких отрицательных чисел она начинается.

г) Определить, сколько раз в этой последовательности меняется знак.

д) Определить, сколько из них больше своих соседей.

е) Найти количество чисел в наиболее длинной последовательности подряд идущих нулей.

ж) Найти, сколько из них принимают наибольшее значение.

5.8. Составить программу, которая выводит на экран коды вводимых символов. Программа заканчивает работу при нажатии клавиши {ESC}, код символа которой равен 27.

5.9. Дан текст произвольной длины, продолжающийся до конца файла. Составить программы для решения следующих задач.

а) Подсчитать общее количество символов и количество пробелов в тексте.

б) Подсчитать количество цифр и количество латинских букв в тексте.

в) Подсчитать количество строк в тексте.

г) Найти порядковый номер первой запятой.

д) Найти порядковый номер последней запятой.

е) Найти количество символов до первой запятой.

ж) Определить, сколько раз в тексте встречается сочетание символов ":=".

з) Текст содержит выражение со скобками. Определить максимальное число уровней вложенности скобок.

и) Выяснить, имеется ли в заданном тексте пара соседних символов “НО” или “ОН”.

к) Выяснить, имеется ли в заданном тексте пара соседних одинаковых символов.

5.10. Входной текст продолжается до конца файла и представляет собой последовательность слов, разделенных произвольным числом пробелов. Составить программы определения следующих величин:

а) количества слов в тексте;

б) количества слов, начинающихся с буквы ‘A’;

в) количества слов, оканчивающихся буквой ‘W’;

г) количества слов, начинающихся и оканчивающихся одной и той же буквой;

д) количества слов, содержащих заданную букву;

е) количества слов, имеющих длину больше трех, но меньше семи символов.

ж) максимальной длины слова.

5.11. Дано целое N>0 и последовательность из N пар вещественных чисел. Каждая пара чисел задает координаты левого и правого конца отрезка, расположенного на оси координат и имеющего ненулевую длину. Найти длину общей части всех отрезков. Если отрезки не имеют общей части, то результатом считать число 0.

5.12. Дано целое N>0 и последовательность из N пар вещественных чисел, задающих координаты на плоскости последовательных вершин ломаной линии. Найти длину ломаной.