2.4. Операторы цикла. Операторы безусловного перехода

Практически все алгоритмы решения задач содержат циклически повторяемые участки. Цикл – это одно из фундаментальных понятий программирования. Под циклом понимается организованное повторение некоторой последовательности операторов.

Любой цикл состоит из кода цикла, т.е. тех операторов, которые выполняются несколько раз, начальных установок, модификации параметра цикла и проверки условия продолжения выполнения цикла.

Один проход цикла называется шагом или итерацией. Проверка условия продолжения цикла происходит на каждой итерации либо до выполнения кода цикла (с предусловием), либо после выполнения (с постусловием).

Для организации циклов используются специальные операторы. Перечень разновидностей операторов цикла языка Си следующий:

– оператор цикла с предусловием;

– оператор цикла с постусловием;

– оператор цикла с предусловием и коррекцией.

Оператор с предусловием while

Цикл с предусловием реализует структурную схему, приведенную на рис. 7.1, *а*, и имеет вид

while* (выражение***)**
код цикла;

Выражение определяет условие повторения кода цикла, представленного простым или составным оператором.

Если выражение в скобках – истина (не равно 0), то выполняется *код цикла*. Это повторяется до тех пор, пока выражение не примет значение 0 (ложь). В этом случае происходит выход из цикла и выполняется оператор, следующий за конструкцией *while*. Если выражение в скобках изначально ложно (т.е. равно 0), то цикл не выполнится ни разу.

Код цикла может включать любое количество операторов, связанных с конструкцией *while*, которые нужно заключить в фигурные скобки (организовать блок), если их более одного.

Переменные, изменяющиеся в коде цикла и используемые при проверке условия продолжения, называются параметрами цикла. Целочисленные параметры цикла, изменяющиеся с постоянным шагом на каждой итерации, называются счетчиками цикла.

Начальные установки могут явно не присутствовать в программе, их смысл состоит в том, чтобы до входа в цикл задать значения переменным, которые в этом цикле используются.

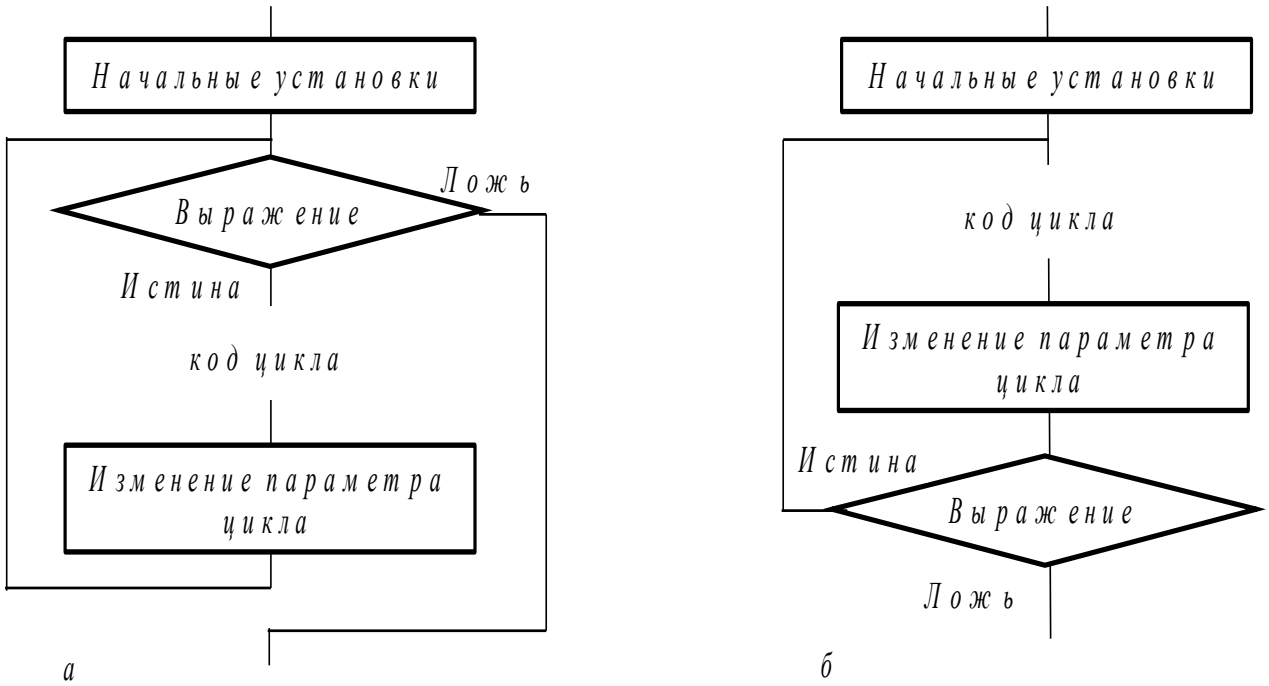

Рис. 7.1. Схемы операторов цикла:
а – цикл с предусловием; *б* – цикл с постусловием

Цикл завершается, если условие его продолжения не выполняется. Возможно принудительное завершение как текущей итерации, так и цикла в целом.

Для этого используют оператор ***continue*** – переход к следующей итерации цикла и ***break*** – выход из цикла (см. разд. 9.2, 9.3).

Передавать управление извне внутрь цикла не рекомендуется, так как получите непредсказуемый результат.

Например, необходимо сосчитать количество символов в строке. Предполагается, что входной поток настроен на начало строки. Тогда подсчет символов выполняется следующим образом:

```
int count = 0;
char ch = getchar();
while ( ch != \n') {
      count++;
      ch = getchar();
 }
```

В языке Си в выражение, управляющее циклом, можно включить и оператор присваивания переменной *ch*, например:

```
char ch;
int count = 0;
while (( ch=getchar()) != '\n') count++;
```

Как видим, переменная *ch* применяется только в выражении, управляющем циклом, поэтому от ch можно отказаться:

```
int count = 0;
while ( getchar() !='\n') count ++;
```

Полезные примеры

1. Организация выхода из бесконечного цикла по нажатии клавиши *Esc*

```
while (1) {                    // Бесконечный цикл
        ...
    if (kbhit() && getch()==27 ) break;
        ...
}
```

Функция ***kbhit***() возвращает значение > 0, если нажата любая клавиша, а функция *getch*() возвращает код нажатой клавиши (код клавиши *Esc* равен 27). В результате выполнения оператора *if*, если будет нажата клавиша *Esc*, выполнится оператор *break* и произойдет выход из цикла.

Приведенный пример – распространенный прием программирования.

2. Организации паузы в работе программы с помощью цикла, выполняющегося до тех пор, пока не нажата любая клавиша

```
        ...
    while (!kbhit());
        ...
```

Оператор цикла с постусловием do – while

Цикл с постусловием реализует структурную схему, приведенную на рис. 7.1, *б*.

Общий вид записи такой конструкции

do
    код цикла;
while (*выражение*);

Код цикла будет выполняться до тех пор, пока *выражение* истинно. Все, что говорилось выше, справедливо и здесь, за исключением того, что данный цикл всегда выполняется хотя бы один раз, даже если изначально выражение ложно.

Здесь сначала выполняется код цикла, после чего проверяется, надо ли его выполнять еще раз.

Следующая программа будет «вас приветствовать» до тех пор, пока будем вводить символ *Y* или *y* (*Yes*). После введения любого другого символа цикл завершит свою работу.

```
#include <stdio.h>
void main(void)
{
    char answer;
    do {
        puts(" Hello! => ");
```

```
        scanf(" %c ", &answer);
    }
    while ((answer=='y')||(answer=='Y'));
}
```

Результат выполнения программы:

Hello! => Y
Hello! => y
Hello! => d

Оператор цикла с предусловием и коррекцией for

Общий вид оператора:

for (*выражение* 1; *выражение* 2; *выражение* 3)
код цикла;

где *выражение* 1 – инициализация счетчика (параметр цикла);
выражение 2 – условие продолжения счета;
выражение 3 – коррекция счетчика.

На рис. 7.2, *а* представлена схема работы цикла *for*, а на рис. 7.2, *б* – символ блок-схемы, использующийся для его обозначения.

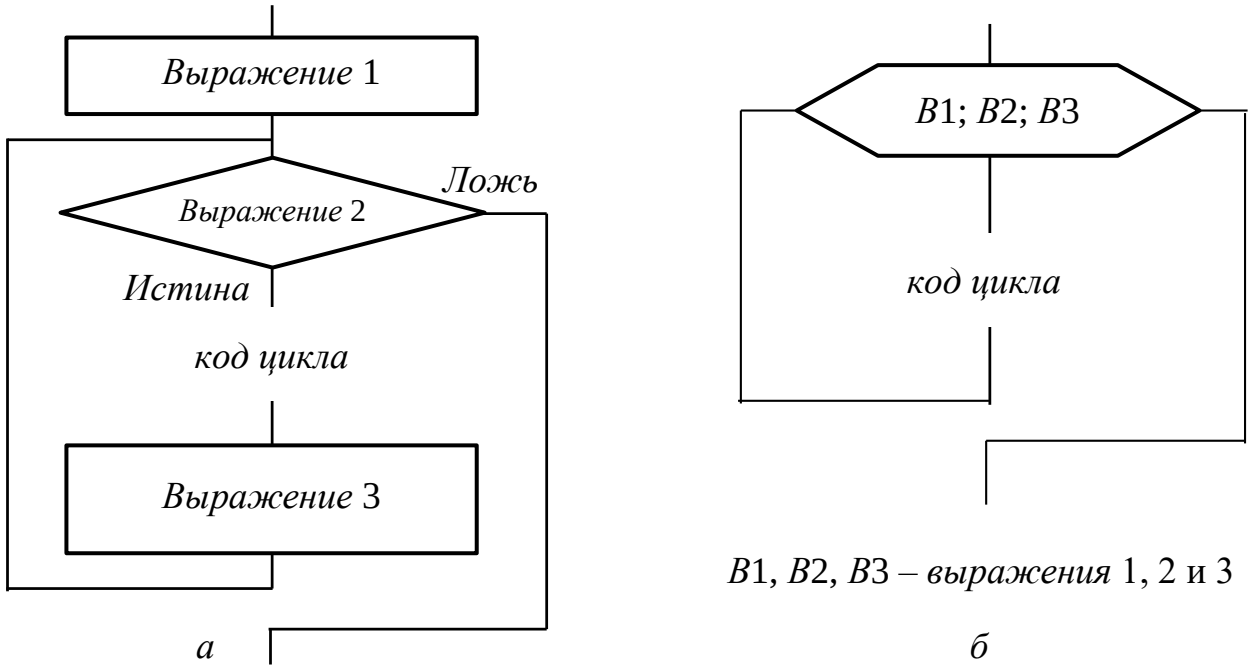

Рис. 7.2. Схемы оператора цикла *for:*
а – схема работы; *б* – блок-схема

Инициализация используется для присвоения счетчику (параметру цикла) начального значения.

Выражение 2 определяет условие выполнения цикла. Как и в предыдущих случаях, если его результат не нулевой («истина»), – то цикл выполняется, иначе – происходит выход из цикла.

Коррекция выполняется после каждой итерации цикла и служит для изменения параметра цикла.

Выражения 1, 2 и 3 могут отсутствовать (пустые выражения), но символы «;» опускать нельзя.

Например, для суммирования первых *N* натуральных чисел можно записать такой код:

```
sum = 0;
for ( i = 1; i<=N; i++)  sum+=i;
```

Заметим, что в *выражении* 1 переменную-счетчик можно декларировать. Например:

```
for (int i = 1; i<=N; i++)
```

Областью действия такой переменной будет код цикла.

Но в старых версиях компиляторов такие действия могут интерпретироваться иначе.

Цикл *for* эквивалентен последовательности инструкций:

```
выражение 1;
while (выражение 2) {
            ...
    выражение 3;
}
```

а оператор

```
for (; выражение 2; )
        код цикла;
```

эквивалентен оператору

```
while (выражение 2)
        код цикла;
```

Если пропущено *выражение* 2, то цикл будет выполняться бесконечно, поскольку пустое условие всегда остается истинным. Бесконечный оператор:

```
for ( ; ; ) код цикла;
```

эквивалентен оператору

```
while (1) код цикла;
```

В заголовке оператора *for* может использоваться операция «запятая». Она позволяет включать в его выражения несколько операторов. Тогда рассмотренный пример суммирования первых *N* натуральных чисел можно записать в следующем виде:

```
for ( sum = 0 , i = 1; i<=N; sum+= i , i++) ;
```

Оператор *for* имеет следующие возможности:

– можно вести подсчет с помощью символов, а не только чисел:

```
for (ch = 'a'; ch <= 'z'; ch++) ... ;
```

– можно проверить выполнение некоторого произвольного условия:

```
for (n = 0; s[i] >= '0' && s[i] < '9'; i++) ... ;
```

или

```
for (n = 1; n*n*n <= 216; n++) ... ;
```

Первое выражение необязательно должно инициализировать переменную. Необходимо только помнить, что первое выражение вычисляется только один раз, перед тем как остальные части начнут выполняться.

```
for (printf(" вводить числа по порядку! \n"); num!=6;)
        scanf("%d", & num);
printf(" последнее число – это то, что нужно. \n");
```

В этом фрагменте первое сообщение выводится на печать один раз, а затем осуществляется прием вводимых чисел, пока не поступит число 6.

Переменные, входящие в *выражения* 2 и 3, можно изменять при выполнении кода цикла, например, значения *k* и *delta*:

```
for (n = 1; n < 10*k; n += delta) ... ;
```

Использование условных выражений позволяет во многих случаях значительно упростить программу, например:

```
for (i = 0; i<n; i++)
        printf("%6d%c", a[i],( (i%10==0) || (i==n–1) ) ? '\n' : ′ ′);
```

В этом цикле печатаются *n* элементов массива *a* по 10 в строке, разделяя каждый столбец одним пробелом и заканчивая каждую строку (включая последнюю) одним символом перевода строки. Символ перевода строки записывается после каждого десятого и *n*-го элементов. За всеми остальными – пробел.

Наиболее часто встречающиеся ***ошибки*** при создании циклов – это использование в коде цикла неинициализированных переменных и неверная запись условия выхода из цикла.

Чтобы избежать ошибок, нужно стараться:

– проверить, всем ли переменным, встречающимся в правой части операторов присваивания в коде цикла, присвоены до этого начальные значения (а также возможно ли выполнение других операторов);

– проверить, изменяется ли в цикле хотя бы одна переменная, входящая в условие выхода из цикла;

– предусмотреть аварийный выход из цикла по достижении некоторого количества итераций;

– если в состав цикла входит не один, а несколько операторов, нужно заключать их в фигурные скобки.

Формально к операторам передачи управления относятся:

– оператор безусловного перехода ***goto***;

– оператор перехода к следующему шагу (итерации) цикла ***continue***;

– выход из цикла, либо оператора *switch* – ***break***;

– оператор возврата из функции ***return***.

Оператор безусловного перехода goto

В языке Си предусмотрен оператор *goto*, общий вид которого

goto *метка* ;

Он предназначен для передачи управления оператору, помеченному указанной *меткой*. Метка представляет собой идентификатор, оформленный по всем правилам идентификации переменных с символом «двоеточие» после него, например, пустой помеченный меткой *m*1 оператор:

```
m1:   ;
```

Область действия метки – функция, где эта метка определена. В случае необходимости можно использовать блок.

Циклы и переключатели можно вкладывать вдруг в друга и наиболее характерный оправданный случай использования оператора *goto* – выполнение прерывания (организация выхода) во вложенной структуре. Например, при возникновении грубых неисправимых ошибок необходимо выйти из двух (или более) вложенных структур (где нельзя использовать непосредственно оператор *break*, т.к. он прерывает только самый внутренний цикл):

```
for (...)
        for (...) {
                ...
                if (ошибка)  goto error;
        }
                ...
      error:  операторы для устранения ошибки;
```

Второй оправданный случай: организация переходов из нескольких мест функции в одно, например, когда перед завершением работы функции необходимо сделать одну и ту же операцию.

Операторы continue, break и return

Оператор ***continue*** может использоваться во всех типах циклов (но не в операторе-переключателе *switch*). Наличие оператора *continue* вызывает пропуск «оставшейся» части итерации и переход к началу следующей, т.е. досрочное завершение текущего шага и переход к следующему шагу.

В циклах *while* и *do-while* это означает непосредственный переход к проверочной части. В цикле *for* управление передается на шаг коррекции, т.е. модификации *выражения* 3.

Оператор *continue* часто используется, когда последующая часть цикла оказывается слишком сложной, так что рассмотрение условия, обратного проверяемому, приводит к слишком высокому уровню вложенности программы.

Оператор ***break*** производит досрочный выход из цикла или оператора-переключателя *switch*, к которому он принадлежит, и передает управление первому оператору, следующему за текущим оператором. То есть *break* обеспечивает переход в точку кода программы, находящуюся за оператором, внутри которого он (*break*) находится.

Оператор ***return*** производит досрочный выход из текущей функции. Он также возвращает значение результата функции:

return *выражение*;

Выражение должно иметь скалярный тип. Правила использования данного оператора будут рассмотрены в гл. 12.

Функции exit и abort

Функция ***exit*** выполняет прерывание программы и используется для нормального, корректного завершения работы программы при возникновении какой-либо внештатной ситуации, например, ошибка при открытии файла (гл. 14). При этом записываются все буферы в соответствующие файлы, закрываются все потоки и вызываются все зарегистрированные стандартные функции завершения.

Прототип этой функции приведен в заголовочном файле *stdlib.h* и выглядит так:

void ***exit*** (*int* ***exit_code*****);**

Параметр данной функции – ненулевое целое число, передаваемое системе программирования (служебное сообщение о возникшей внештатной ситуации).

Для завершения работы программы также может использоваться функция

void ***abort*** (*void*);

действия которой аналогичны функции *exit*(3).

Советы по программированию

При выполнении вариантов заданий придерживайтесь следующих ключевых моментов.

1. Выражение, стоящее в круглых скобках операторов *if*, *while* и *do-while*, вычисляется по правилам стандартных приоритетов операций.

2. Если в какой-либо ветви вычислений условного оператора или в цикле требуется выполнить два (и более) оператора, то они при помощи фигурных скобок объединяются в блок.

3. Проверка вещественных величин на равенство, как правило, из-за ограниченной разрядности дает неверный результат.

4. Чтобы получить максимальную читаемость и простоту структуры программы, надо правильно выбирать способ реализации ветвлений (с помощью *if*, *switch*, или условных операций), а также наиболее подходящий оператор цикла.

5. Выражение в операторе *switch* и константные выражения в *case* должны быть целочисленного или символьного типов.

6. Рекомендуется использовать в операторе *switch* ветвь *default*.

7. После каждой ветви для передачи управления на точку кода за оператором *switch* используется оператор *break*.

8. При построении любого цикла надо не забывать тот факт, что в нем всегда явно или неявно присутствуют четыре основных элемента: начальные

установки, код цикла, модификация параметра цикла и проверка условия на продолжение цикла.

9. Если количество повторений цикла заранее не известно (реализуется итерационный процесс), необходимо предусмотреть аварийное завершение цикла при получении достаточно большого количества итераций.

10. При использовании бесконечного цикла обязательно необходима организация выхода из цикла по условию.

ЗАДАНИЕ 2. Разветвляющиеся алгоритмы

Первый уровень сложности

Составить программу нахождения требуемого значения с исходными данными *x*, *y*, *z*. Обозначение: *min* и *max* – нахождение минимального и максимального из перечисленных в скобках значений элементов.

1. $m=\dfrac{\max(x,y,z)}{\min(x,y)}+5$;

2. $n=\dfrac{\min(x+y,y-z)}{\max(x,y)}$;

3. $p=\dfrac{|\min(x,y)-\max(y,z)|}{2}$;

4. $q=\dfrac{\max(x+y+z,\ x\cdot y\cdot z)}{\min(x+y+z,\ x\cdot y\cdot z)}$;

5. $r=\dfrac{\max[\min(x,y),z]}{3}$;

6. $s=\dfrac{\min[\max(x,y),\max(y,z)]}{\max(y,z)}$;

7. $t=\dfrac{\max[\min(x,5),\max(y,0)]}{5}$;

8. $v=\max[\min(x-y,y-x),0]$;

9. $w=\max^2[\max(x\cdot y,x+y),0]$;

10. $z=\dfrac{\min(0,x)\ -\min(0,y)}{\max^2(y,x)}$;

11. $u=\dfrac{\min(y,z)}{\max[\min(x,y),\min(y,z)]}$;

12. $q=\dfrac{\min(x+y+z,\ x\cdot y\cdot z)}{\min(x-y+z,\ x\cdot y/z)}$;

13. $q=\dfrac{\max(x+y+z,\ x\cdot y\cdot z)}{\max[x+y+z,\ x/(y\cdot z)]}$;

14. $u=\dfrac{\max(y,z)}{\min[\min(x,y),\min(y,z)]}$;

15. $u=\dfrac{\min(y,z)}{\max[\max(x,y),\max(y,z)]}$.

Второй уровень сложности

Вычислить значение *y* в зависимости от выбранной функции φ(*x*), аргумент которой определяется из поставленного условия. Возможные значения функции

$\varphi(x)$: $2x$, x^2, $x/3$. Предусмотреть вывод сообщений, показывающих, при каком условии и с какой функцией производились вычисления y.

1. $y = a\ln(1 + x^{1/5}) + \cos^2[\varphi(x) + 1]$, где $x = \begin{cases} z^2; & z < 1; \\ z + 1; & z \ge 1. \end{cases}$

2. $y = \dfrac{2a\varphi(x) + b\cos\sqrt{|x|}}{x^2 + 5}$, где $x = \begin{cases} 2 + z; & z < 1; \\ \sin^2 z; & z \ge 1. \end{cases}$

3. $y = -\pi\varphi(x) + a\cos^2 x^3 + b\sin^3 x^2$, где $x = \begin{cases} z; & z < 1; \\ \sqrt{z^3}; & z \ge 1. \end{cases}$

4. $y = 2a\cos^3 x^2 + \sin^2 x^3 - b\varphi(x)$, где $x = \begin{cases} z^3 + 0{,}2; & z < 1; \\ z + \ln z; & z \ge 1. \end{cases}$

5. $y = a\varphi(x) - \ln(x + 2{,}5) + b(e^x - e^{-x})$, где $x = \begin{cases} -z/3; & z < -1; \\ |z|; & z \ge -1. \end{cases}$

6. $y = \dfrac{2}{3}a\sin^2 x - \dfrac{3b}{4}\cos^2\varphi(x)$, где $x = \begin{cases} z; & z < 0; \\ \sin z; & z \ge 0. \end{cases}$

7. $y = \sin^3[c\varphi(x) + d^2 + x^2]$, где $x = \begin{cases} z^2 - z; & z < 0; \\ z^3; & z \ge 0. \end{cases}$

8. $y = \sin^2\varphi(x) + a\cos^5 x^3 + c\ln x^{2/5}$, где $x = \begin{cases} 2z + 1; & z \ge 0; \\ \ln(z^2 - z); & z < 0. \end{cases}$

9. $y = \dfrac{b\varphi(x)}{\cos x} + a\ln\left|tg\dfrac{x}{2}\right|$, где $x = \begin{cases} z^2/2; & z \le 0; \\ \sqrt{z}; & z > 0. \end{cases}$

10. $y = \dfrac{d\varphi(x)e^{\sin^3 x} + c\ln(x + 1)}{\sqrt{x}}$, где $x = \begin{cases} z^2 + 1; & z < 1; \\ z - 1; & z \ge 1; \end{cases}$

11. $y = \dfrac{2{,}5a \cdot e^{-3x} - 4bx^2}{\ln|x| + \varphi(x)}$, где $x = \begin{cases} \dfrac{1}{z^2 + 2z}; & z > 0; \\ 1 - z^3; & z \le 0. \end{cases}$

12. $y = a\sin^3[\varphi(x)^2 - 1] + c\ln|x| + e^x$, где $x = \begin{cases} z^2 + 1; & z \le 1; \\ 1/\sqrt{z - 1}; & z > 1. \end{cases}$

13. $y = \sin[n\varphi(x)] + \cos kx + \ln mx$, где $x = \begin{cases} z; & z > 1; \\ z^2 + 1; & z \le 1. \end{cases}$

14. $y = b\cos[a\varphi(x)] + \sin\dfrac{x}{5} + ae^x$, где $x = \begin{cases} \sqrt{z}; & z > 0; \\ 3z + 1; & z \le 0. \end{cases}$

15. $y = 2\varphi(x)[a\sin x + d \cdot e^{-(x+3)}]$, где $x = \begin{cases} -3z; & z > 0; \\ z^2; & z \le 0. \end{cases}$

ЗАДАНИЕ 3. Циклические алгоритмы

Первый уровень сложности

Составить программу для определения таблицы значений функции ***y*** в произвольном диапазоне [***a***, ***b***] изменения аргумента ***x*** с произвольным шагом ***h***. Значения ***a, b, h*** вводятся с клавиатуры. Таблица должна содержать следующие столбцы: порядковый номер, значение аргумента ***x***, значение функции, сообщение о возрастании или убывании функции.

Определить максимальное и минимальное значения функции.

1. $Y(x) = \dfrac{2\sin x}{(1-x)^2}$, $a = -\pi$; $b = \pi$; $h = 0{,}4$.
2. $Y(x) = -\ln\left|2\sin\dfrac{x}{2}\right|$, $a = 0{,}7$; $b = 1{,}8$; $h = 0{,}1$.
3. $Y(x) = \dfrac{x\sin(\frac{\pi}{4})}{1-2x\cos\frac{\pi}{4}+x^2}$, $a = -0{,}5$; $b = 2{,}5$; $h = 0{,}2$.
4. $Y(x) = (1-\dfrac{x^2}{4})\cos x - \dfrac{x}{2}\sin x$, $a = -0{,}9$; $b = 2{,}7$; $h = 0{,}3$.
5. $Y(x) = \dfrac{x\cos\frac{\pi}{4} - x^2}{1-2x\cos\frac{\pi}{4}+x^2}$, $a = -2$; $b = 0{,}8$; $h = 0{,}2$.
6. $Y(x) = (\dfrac{x^2}{4} + \dfrac{x}{2} - 3)\cdot e^{\frac{x}{2}}$, $a = -1{,}9$; $b = 2{,}7$; $h = 0{,}3$.
7. $Y(x) = x^2\sqrt{15 + 10\sin(x+\pi)}$, $a = -0{,}4\pi$; $b = 0{,}4\pi$; $h = 0{,}5$.
8. $Y(x) = e^x \sin x$, $a = -0{,}3\pi$; $b = 1{,}3\pi$; $h = \pi/10$.
9. $Y(x) = x^2 \cos x \sin x$, $a = -\pi/2$; $b = \pi/2$; $h = \pi/10$.
10. $Y(x) = x\log(|x - 0{,}6|)$, $a = -3$; $b = 3$; $h = 0{,}5$.
11. $Y(x) = \dfrac{x}{2}\cos x - \sin x$, $a = -\pi$; $b = \pi$; $h = \pi/6$.
12. $Y(x) = e^x + \sqrt{1 + e^{2x}} - 2$, $a = -0{,}9$; $b = 1$, $h = 0{,}3$.
13. $Y(x) = (1-\dfrac{x^2}{4})\cos x - \dfrac{x}{2}\sin x$, $a = -0{,}9$; $b = 2{,}7$; $h = 0{,}3$.

14. $Y(x)=\dfrac{1}{x^2-x+1}$, a = –0,1; b = 2; h = 0,1.

15. $Y(x)=\dfrac{\sin x\cos x}{\sqrt{x}}$, a = π; b = 2π; h = π/15.

Второй уровень сложности

Значение аргумента ***x*** изменяется от ***a*** до ***b*** с шагом ***h***. Для каждого ***x*** найти значения функции $Y(x)$, суммы $S(x)$ и $|Y(x)–S(x)|$ и вывести в виде таблицы. Значения a, b, h и n вводятся с клавиатуры. Так как значение $S(x)$ является рядом разложения функции $Y(x)$, значения S и Y для заданного аргумента x должны совпадать в целой части и в первых двух-четырех позициях после десятичной точки.

Работу программы проверить для a = 0,1; b = 1,0; h = 0,1; значение параметра n выбрать в зависимости от задания.

1. $S(x)=\sum_{k=0}^{n}(-1)^k\dfrac{x^{2k+1}}{(2k+1)!}$, $Y(x)=\sin(x)$.

2. $S(x)=\sum_{k=1}^{n}(-1)^{k+1}\dfrac{\sin(kx)}{k}$, $Y(x)=\dfrac{x}{2}$.

3. $S(x)=\sum_{k=0}^{n}\dfrac{\cos(\frac{k\pi}{4})}{k!}x^k$, $Y(x)=e^{x\cos\frac{\pi}{4}}\cos(x\sin(\pi/4))$.

4. $S(x)=\sum_{k=0}^{n}(-1)^k\dfrac{x^{2k}}{(2k)!}$, $Y(x)=\cos(x)$.

5. $S(x)=\sum_{k=0}^{n}\dfrac{\cos(kx)}{k!}$, $Y(x)=e^{\cos x}\cos(\sin(x))$.

6. $S(x)=\sum_{k=0}^{n}\dfrac{2k+1}{k!}x^{2k}$, $Y(x)=(1+2x^2)e^{x^2}$.

7. $S(x)=\sum_{k=1}^{n}\dfrac{x^k\cos\frac{k\pi}{3}}{k}$, $Y(x)=-\dfrac{1}{2}\ln(1-2x\cos\dfrac{\pi}{3}+x^2)$.

8. $S(x)=\sum_{k=1}^{n}(-1)^k\dfrac{\cos(kx)}{k^2}$, $Y(x)=\dfrac{1}{4}(x^2-\pi^2/3)$.

9. $S(x)=\sum_{k=1}^{n}(-1)^{k+1}\dfrac{x^{2k+1}}{4k^2-1}$, $Y(x)=\dfrac{1+x^2}{2}arctg(x)-x/2$.

10. $S(x)=\sum_{k=0}^{n}\dfrac{x^{2k}}{(2k)!}$, $Y(x)=\dfrac{e^x+e^{-x}}{2}$.

11. $S(x)=\sum_{k=0}^{n}\frac{k^2+1}{k!}(x/2)^k$, $Y(x)=(x^2/4+x/2+1)e^{x/2}$.

12. $S(x)=\sum_{k=0}^{n}(-1)^k\frac{2k^2+1}{(2k)!}x^{2k}$, $Y(x)=(1-\frac{x^2}{2})\cos(x)-\frac{x}{2}\sin(x)$.

13. $S(x)=\sum_{k=1}^{n}(-1)^k\frac{(2x)^{2k}}{(2k)!}$, $Y(x)=2(\cos^2 x-1)$.

14. $S(x)=\sum_{k=0}^{n}\frac{x^{2k+1}}{(2k+1)!}$, $Y(x)=\frac{e^x-e^{-x}}{2}$.

15. $S(x)=\sum_{k=1}^{n}(-1)^{k+1}\frac{x^{2k}}{2k(2k-1)}$, $Y(x)=xarctg(x)-\ln\sqrt{1+x^2}$.