ЛЕКЦИИ 9 – 10

Операторы управления
Оператор ветвления
Оператор выбора
Операторы циклов
Операторы безусловных переходов

Виды операторов управления

Во всех языках программирования высокого уровня выделяют три основные группы операторов управления:

- операторы условия,
- операторы циклов,
- безусловные операторы.

Составной оператор

Составной оператор языка С – оператор, состоящий из последовательности нескольких простых и (или) составных операторов, заключенных в фигурные скобки. Также составной оператор называют блоком операторов.

Составной оператор

Пример:

```
{
  scanf(“%f”,&a[i]);
  printf(“a[%d]=%f\n”,i,a[i]);
}
```

Составной оператор может также включать в себя и другие составные операторы. Точка с запятой после закрывающей фигурной скобки не ставится.

Оператор ветвления

Оператор ветвления (условия) – оператор управления, осуществляющий выполнение одного и только одного составного оператора из двух возможных в зависимости от условия.

Синтаксис оператора ветвления:

if(условие) составной оператор №1

else составной оператор №2

Оператор ветвления

Если составной оператор в основной или альтернативной ветви оператора управления содержит только один простой оператор, то фигурные скобки можно опустить. В таком случае синтаксис оператора ветвления примет вид:

if(условие) простой оператор №1;
else простой оператор №2;

Оператор ветвления

Основное предназначение оператора ветвления в языке С – реализация разветвляющихся алгоритмов.
Например, фрагмент программы, для вычисления функции:

$$Y(x) = \begin{cases} x^2, & x < 0 \\ 2 \cdot x, & x \geq 0 \end{cases}$$

```
double x,y;
printf("Введите значение X: ");
scanf("%lf",&x);
if(x<0.0) y = x*x;
  else y = 2.0*x;
printf("Результат: %lf\n",y);
```

Оператор ветвления

При построении программ с несколькими подряд идущими операторами ветвления необходимо учитывать следующее правило: оператор **else** относится к последнему оператору **if**. Например, рассмотрим фрагмент программы:

```
if(x>10.0) y = 2*x+5;
if(y<z) y = fabs(x);
  else y = 2.5*x;
```

Оператор ветвления

Если необходимо нарушить этот порядок (т.е. ветвь **else** отнести к первому оператору **if**), то необходимо второй оператор **if** включить в составной оператор прямой ветви первого оператора **if**:

```
if(x>10.0){
  y = 2*x+5;
  if(y<z) z = fabs(x);
} else y = 2.5*x;
```

Оператор ветвления

В программировании часто встречается необходимость создания каскадных операторов ветвления. Синтаксис такой структуры имеет вид:

if(условие № 1) составной оператор № 1
else if(условие № 2) составной оператор № 2
...
else if(условие № N) составной оператор № N
else составной оператор № N+1

Оператор ветвления

Например, фрагмент программы, для вычисления функции:

$$Y(x) = \begin{cases} x^2, x < 0 \\ 2 \cdot x, 0 \le x < 5 \\ x + 5, x \ge 5 \end{cases}$$

```
double x,y;
printf("Введите значение X: ");
scanf("%lf",&x);
if(x<0.0) y = x*x;
  else if(x<5) y = 2.0*x;
    else y = x + 5.0;
printf("Результат: %lf\n",y);
```

Правила форматирования

Альтернативная ветвь оператора ветвления всегда указывается на новой строке с отступом:

if(...) ...
  else ...

Если в прямой или альтернативной ветви оператора ветвления находятся один простой оператор, то он указывается на той же строке:

if(...) оператор № 1;
  else оператор № 2;

Правила форматирования

Если в прямой или альтернативной ветви оператора ветвления находятся составные операторы, то следует придерживаться следующего форматирования:

```
if(условие){
  оператор № 1;
  ...
  оператор № N;
}else{
  оператор № N+1;
  ...
}
```

Правила оформления

ИСКЛЮЧЕНИЕ: если операторы в прямой и альтернативной ветви имеют краткую запись, то допускается оператор ветвления записывать в одну строчку:

if(условие) оператор №1; **else** оператор №2;

Пример

Квадратное уравнение вида $A \cdot x^2 + B \cdot x + C = 0$ задается коэффициентами A, B и C. Определить корни уравнения.

```
#include <stdio.h>
#include <math.h>
int main(int argc, char *argv[])
{
  double A,B,C;
  printf("Введите коэффициенты A, B и C: "); scanf("%lf %lf %lf",&A,&B,&C);
  double D = B*B - 4.0*A*C;
  if(D < 0.0){
    printf("Нет действительных корней уравнения!\n");
  }else if(D > 0.0){
    printf("Корни уравнения: %.4lf\t%.4lf\n",
            (-B-sqrt(D))/(2.0*A), (-B+sqrt(D))/(2.0*A));
  }else{
     printf("Один корень: %.4lf\n",-B/(2.0*A));
  }
  return 0;
}
```

Оператор выбора

Необходимо реализовать программу: дан номер дня недели (1 – понедельник, 2 – вторник и т.д.), необходимо вывести на экран название этого дня. При каскадировании операторов ветвления эта программа будет иметь вид:

```
int n;
printf(“Введите номер дня недели:”);
scanf(“%d”,&n);
if(n==1) printf(“Понедельник\n”);
 else if(n==2) printf(“Вторник\n”);
  else if(n==3) printf(“Среда\n”);
    else if(n==4) printf(“Четверг\n”);
     else if(n==5) printf(“Пятница\n”);
      else if(n==6) printf(“Суббота\n”);
       else if(n==7) printf(“Воскресенье\n”);
        else printf(“Неправильный номер!\n”);
```

Оператор выбора

Оператор выбора – оператор управления, осуществляющий выполнение одного или нескольких действий из набора возможных в зависимости от значения некоторого выражения.

Синтаксис оператора выбора:

```
switch(выражение){
  case константа № 1: составной оператор № 1;
  case константа № 2: составной оператор № 2;
  ...
  case константа № N: составной оператор № N;
  default: составной оператор № N+1;
}
```

Оператор выбора

ПРИМЕЧАНИЕ: Выполнение всех составных операторов после первого совпадения не всегда удобно. Поэтому последним оператором в составном операторе обычно является оператор безусловного перехода **break**, который осуществляет завершение выполнения оператора **switch**.

Оператор выбора

Программа о днях недели с использованием оператора выбора будет выглядеть следующим образом:

```
int n;
printf(“Введите номер дня недели: ”);
scanf(“%d”,&n);
switch(n){
  case 1: {printf(“Понедельник\n”); break;}
  case 2: {printf(“Вторник\n”); break;}
  case 3: {printf(“Среда\n”); break;}
  case 4: {printf(“Четверг\n”); break;}
  case 5: {printf(“Пятница\n”); break;}
  case 6: {printf(“Суббота\n”); break;}
  case 7: {printf(“Воскресенье\n”); break;}
  default: printf(“Неправильной номер!\n”);
}
```

Оператор выбора

В некоторых программах возможность выполнения всех последующих блоков очень удобна. Например, пользователь вводит номер дня недели, необходимо вывести на экран сообщение: будний это день или выходной.

```
int n;
printf(“Введите номер дня недели: ”);
scanf(“%d”,&n);
switch(n){
  case 1: case 2: case 3: case 4: case 5:
    {printf(“Будний день!\n”); break;}
  case 6: case 7:
   {printf(“Выходной день!\n”); break;}
  default: printf(“Неправильный номер!\n”);
}
```

Правила форматирования

При форматировании текста программ включающих в себя оператор выбора рекомендуется придерживаться следующих правил:

Общий формат оператора выбора должен быть следующий:

```
switch(...){
  операторы
}
```

Каждый оператор **case** должен начинаться с новой строки с отступом относительно записи **switch**.

Правила форматирования

Если составной оператор после оператора **case** большой, то он записывается несколькими строками в следующем формате:

case константа:**{**
Оператор № 1;
...
Оператор № N;
break;
}

ИСКЛЮЧЕНИЕ: Операторы **case** допускается записывать в одну строку, если они содержат только константы, а составной оператор содержится только в последней из них.

Пример

Пользователь вводит дату не високосного года в формате DD.MM, где DD - день, MM - месяц. Вывести на экран введенную дату словесно. Например, пользователь ввел: 10.05. На экране: десятое мая.

```
#include <stdio.h>
```

Пример (продолжение)

```
int main(int argc, char *argv[])
{
  unsigned day, mon;
  printf("Введите дату: "); scanf("%u.%u",&day,&mon);
  int flag = (day<1)||(day>31)||(mon<1)||(mon>12);
  flag +=
    ((mon==4)||(mon==6)||(mon==9)||(mon==11))&&(day==31);
  flag += (mon==2)&&(day>28);
  if(flag) {printf("Неправильная дата!\n"); return 0;}
  unsigned d1 = day/10, d2 = day%10;
```

Пример (продолжение)

```
switch(d1){
  //Если это тридцатые числа
  case 3:{
    if(d2==0) printf("Тридцатое ");
      else printf("Тридцать первое ");
    break;
   }
  //Если это двадцатые числа
   case 2:{
    if(d2==0) {printf("Двадцатое "); break;}
      else printf("Двадцать ");
   }
   //Если это первые числа
   case 0: {
     switch(d2){
       case 1: {printf("первое "); break;}
       case 2: {printf("второе "); break;}
       …
       case 9: {printf("девятое "); break;}
     }
     break;
   }
```

```
//Если число от десятого до
  девятнадцатого
   case 1:{
     switch(d2){
       case 0: {printf("Десятое "); break;}
       case 1: {printf("Одиннадцатое ");
  break;}
       …
       case 9: {printf("Девятнадцатое ");
  break;}
     }
     break;
   }
}
```

Пример (продолжение)

```
switch(mon){
    case  1: {printf("января"); break;}
    case  2: {printf("февраля"); break;}
    case  3: {printf("марта"); break;}
    case  4: {printf("апреля"); break;}
    case  5: {printf("мая"); break;}
    case  6: {printf("июня"); break;}
    case  7: {printf("июля"); break;}
    case  8: {printf("августа"); break;}
    case  9: {printf("сентября"); break;}
    case 10: {printf("октября"); break;}
    case 11: {printf("ноября"); break;}
    case 12: {printf("декабря"); break;}
  }
  printf("\n");
  return 0;
}
```

Операторы циклов

Операторы циклов предназначены для реализации циклических алгоритмов.

В языках программирования высокого уровня операторы циклов делятся на две группы:

- операторы циклов со счетчиком – выполняются определенное число итераций;
- операторы циклов с условием – выполняются пока условие истинно или ложно.

Операторы циклов

Операторы циклов с условием могут быть классифицированы по следующим признакам:

- с предусловием или с постусловием;
- выполнение пока условие истинно или ложно.

Операторы циклов с предусловием – сначала выполняется проверка условия, а затем тело цикла. В операторах циклов с постусловием наоборот. Циклы с условием могут выполняться пока условие истинно или пока условие ложно. Последний тип таких операторов циклов в языке С отсутствует.

Операторы циклов

В языке С оператор цикла состоит из двух основных частей:

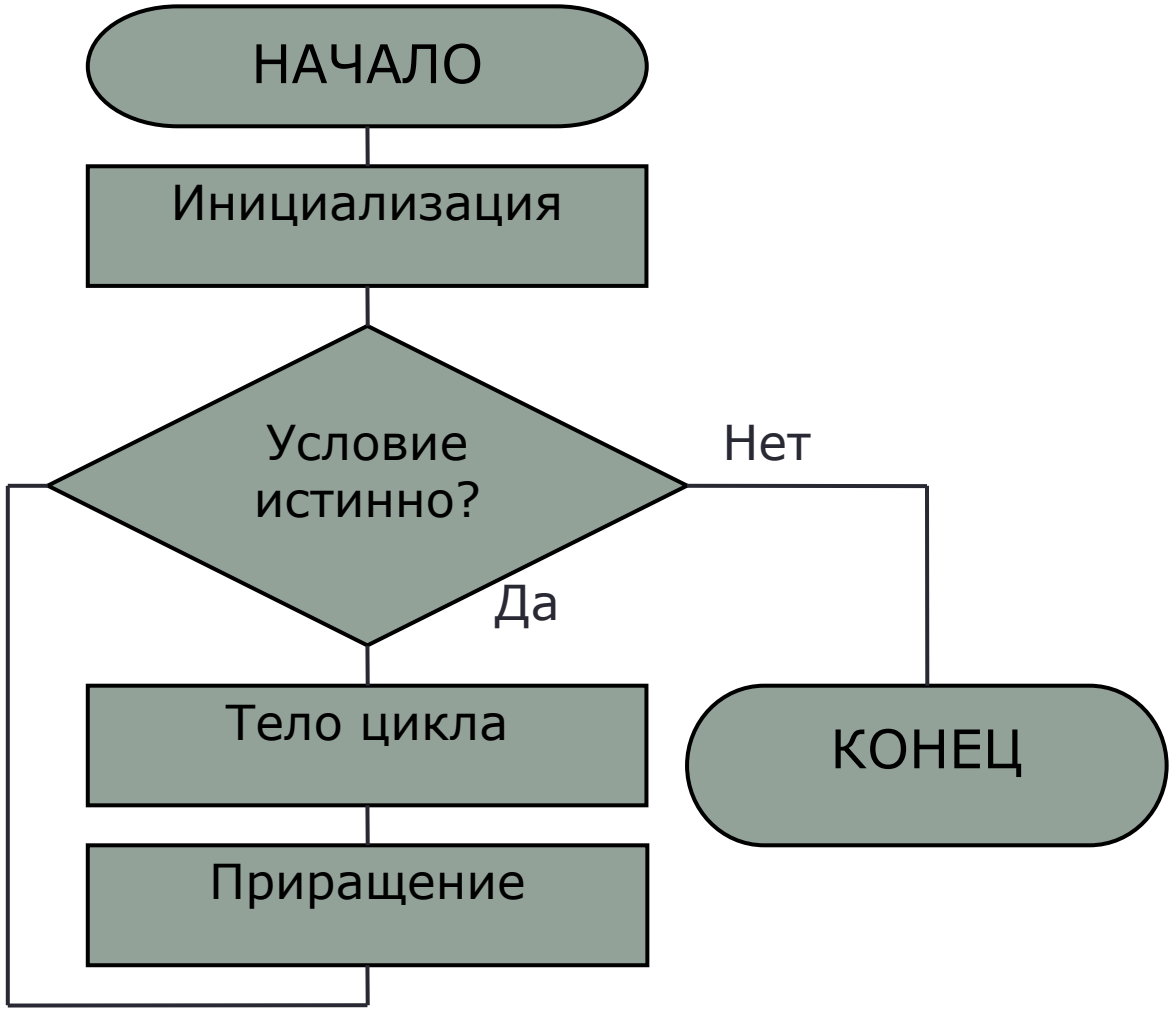
- непосредственно сам оператор цикла (**for**, **while**, **do** ... **while**);
- тело цикла – простой или составной оператор.

Оператор цикла со счетчиком

Оператор цикла со счетчиком предполагает, что количество итераций (выполнений) тела цикла жестко задано в виде некоторой константы или значения переменной. На самом деле данный тип циклов в языках программирования, как правило, значительно более гибок. В языке С оператор цикла со счетчиком имеет следующий синтаксис:

for(инициализация; условие; приращение)
тело цикла

Оператор цикла со счетчиком



Оператор цикла со счетчиком

Например, пользователь вводит последовательность из десяти положительных чисел. Определить максимум этой последовательности.

```
unsigned max = 0;
printf(“Введите последовательность: ”);
for(int i=0;i<10;i++){
  unsigned value;
  scanf(“%u”,&value);
  if(max < value) max = value;
}
printf(“Максимальное значение: %u\n”,max);
```

Оператор цикла со счетчиком

В операторе **for** любой из блоков может отсутствовать. Если отсутствует какой-либо блок в круглых скобках (инициализация, условие или приращение), то разделитель этого блока все равно присутствует. Если отсутствует тело цикла, то после закрывающей оператор **for** круглой скобки просто указывается точка с запятой.

Оператор цикла со счетчиком

Рассмотрим задачу вычисления факториала числа. Любой из ниже приведенных фрагментов программы (циклов) будет правильным:

```
for(int i=1, fact=1.0; i<=n; i++) fact *= i;
for(int i=1, fact=1.0; i<=n;  fact *= i++);
for(int i=1, fact=1.0; i<=n;) fact *= i++;
for(fact=1.0;n>0;) fact *= n--;
```

Оператор цикла со счетчиком

Возможность пропуска блоков в операторе **for** является довольно удобной, хотя в некоторых случаях вообще лишена смысла.

Например, допустима следующая конструкция:

for(;;);

Это вечный цикл, который не выполняет никаких действий.

Правила форматирования

При написании программ с использованием цикла **for** рекомендуется придерживаться следующих правил форматирования текста программы:

Общий формат цикла **for** при составном операторе в теле цикла:

```
for(...){
  оператор № 1;
  ...
  оператор № 2;
}
```

Общий формат цикла **for** при простом операторе в теле цикла:

```
for(...)
  оператор;
```

Правила форматирования

Если тело цикла состоит из простого или составного операторов имеющих достаточно краткую запись, то тело цикла допускается указывать на той же строке, что и сам цикл:

for(...) оператор;

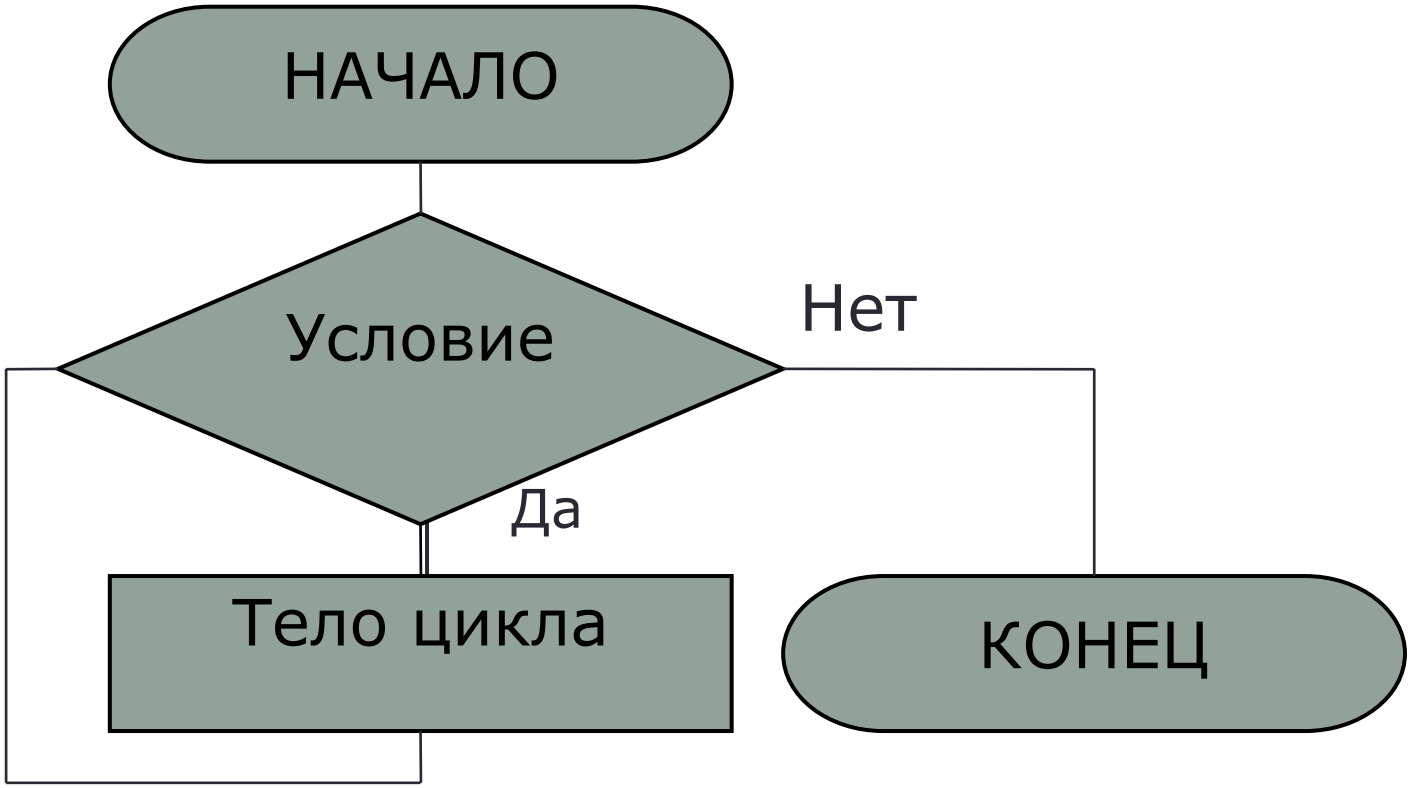
for(...) **{**оператор № 1; ... оператор № N;**}**

Оператор цикла с предусловием

Оператор цикла с предусловием (сначала проверяется условие, а затем выполняется тело цикла) имеет следующий синтаксис:

while(условие) тело цикла

Оператор цикла с предусловием



Оператор цикла с предусловием

Например, в следующем фрагменте программы осуществляется вывод на экран таблицы значений функции синус, если аргумент изменяется от *a* до *b* с шагом *h*. Переменные *a*, *b* и *h* вещественные (тип **double**), и их значения уже введены.

```
while(a <= b){
  printf("sin(%.4lf) = %.4lf\n",a,sin(a));
  a += h;
}
```

Оператор цикла с предусловием

ПРИМЕЧАНИЕ: В языке С данную программу можно также реализовать и с использованием цикла **for**:

```
for(;a<=b;a+=h)
  printf("sin(%.4lf) = %.4lf\n",a,sin(a));
```

Правила форматирования

При написании программ с использованием цикла **while** рекомендуется придерживаться следующих правил форматирования текста программы:

Общий формат цикла **while** при составном операторе в теле цикла:

```
while(условие){
  оператор № 1;
  ...
  оператор № 2;
}
```

Общий формат цикла **while** при простом операторе в теле цикла:

```
while(условие)
  оператор;
```

Правила форматирования

Если тело цикла состоит из простого или составного операторов имеющих достаточно краткую запись, то тело цикла допускается указывать на той же строке, что и сам цикл:

while(условие) оператор;

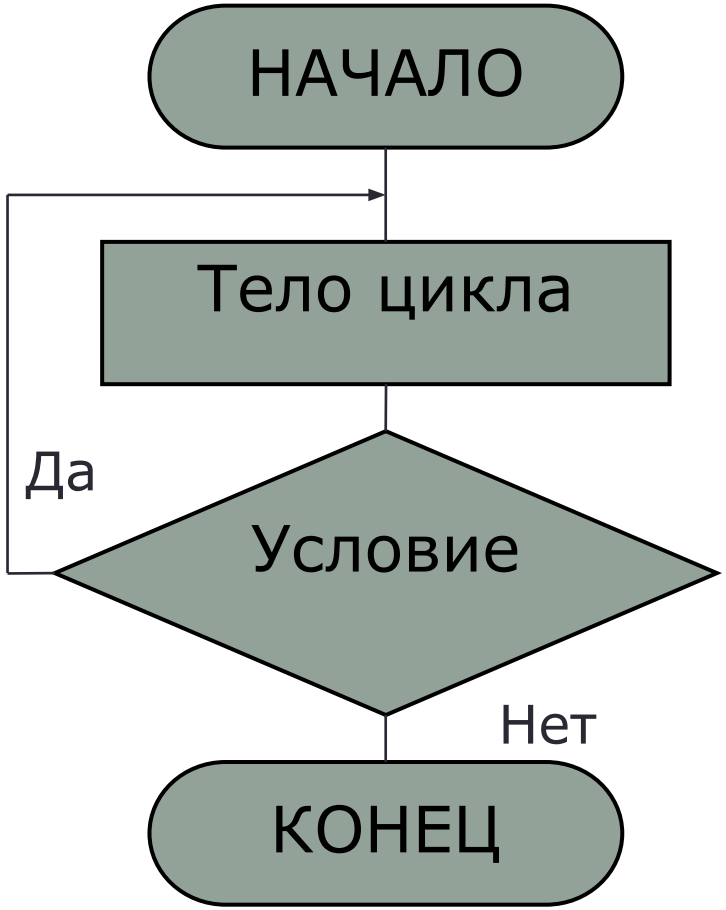
while(условие) **{**оператор № 1; ... оператор № N;**}**

Оператор цикла с постусловием

Оператор цикла с постусловием (сначала выполняется тело цикла, а затем проверяется условие) имеет следующий синтаксис:

do тело цикла **while**(условие);

Оператор цикла с постусловием



Оператор цикла с постусловием

Например, в следующем фрагменте программы осуществляется вычисление суммы ряда с точностью *eps* (вещественная переменная типа **double**, содержит значение погрешности вычисления).

```
double s = 0.0;
unsigned i = 1;
do{
  s = 1.0/i;
  sum += s;
  i++;
}while(eps < s);
```

Правила форматирования

При написании программ с использованием цикла **do...while** рекомендуется придерживаться следующих правил форматирования текста программы:

Общий формат цикла **do...while** при составном операторе в теле цикла:

```
do{
  оператор № 1;
  ...
  оператор № 2;
}while(условие);
```

Общий формат цикла **do...while** при простом операторе в теле цикла:

do оператор **while**(условие);

Правила форматирования

Если тело цикла состоит из составного оператора имеющего достаточно краткую запись, то тело цикла допускается указывать на той же строке, что и сам цикл:

do {оператор №1; ... оператор №N;**} while**(условие);

Операторы безусловных переходов

Операторы безусловного перехода осуществляют управление потоком программы в соответствии с некоторыми правилами без каких-либо условий. В языке С присутствуют следующие операторы безусловных переходов:

- оператор прерывания выполнения тела цикла и переход к следующей итерации;
- оператор прерывания выполнения оператора цикла или оператора выбора и переход к следующему оператору;
- оператор безусловного перехода по метке.

Операторы безусловных переходов

Оператор **continue** – оператор прерывания выполнения тела цикла и переход к следующей итерации.

Рассмотрим использование этого оператора на примере фрагмента программы, вычисляющего сумму трехзначных цифр не кратных пяти или трем.

```
unsigned summa = 0;
for(unsigned n=100;n<1000;n++){
  if((n%5==0)||(n%3==0)) continue;
  summa += n;
}
```

Операторы безусловных переходов

ПРИМЕЧАНИЕ: Использования оператора **continue** можно избежать практически в любом случае. Например, предыдущий фрагмент программы можно преобразовать к виду:

```
unsigned summa = 0;
for(unsigned n=100;n<1000;n++)
  if((n%5!=0)&&(n%3!=0)) summa += n;
```

Операторы безусловных переходов

Оператор **continue** осуществляет прерывание тела только того цикла, внутри которого он находится. Например, во фрагменте программы слева оператор **continue** относится к вложенному циклу, а справа – к внешнему циклу:

```
for(int i=0;i<N;i++){
   ...
   for(int j=0;j<M;j++){
      ...
       if(...) continue;
       ...
   }
}
```

```
for(int i=0;i<N;i++){
   ...
   for(int j=0;j<M;j++){
      ...
   }
   if(...) continue;
}
```

Операторы безусловных переходов

Оператор **break** – оператор прерывания выполнения оператора цикла (**for**, **while**, **do...while**) или оператора выбора (**switch**).

Рассмотрим использование этого оператора на примере фрагмента программы, находящей третье по счету двухзначное число, сумма цифр которого равна семи.

```
unsigned num = 0;
for(unsigned i=10, j=0;i<100;i++){
  if(i%10 + i/10 == 7) j++;
  if(j==3) {num = i; break;}
}
```

Операторы безусловных переходов

ПРИМЕЧАНИЕ: Использования оператора **break** можно избежать практически в любом случае. Например, предыдущий фрагмент программы можно преобразовать к виду:

```
unsigned num = 0;
for(unsigned i=10, j=0;(i<100)&&(j<3);i++){
  if(i%10 + i/10 == 7) j++;
  num = i;
}
```

Операторы безусловных переходов

Оператор **break** осуществляет прерывание только того цикла, внутри которого он находится. Например, во фрагменте программы слева оператор **break** относится к вложенному циклу, а справа – к внешнему циклу:

```
for(int i=0;i<N;i++){
   ...
   for(int j=0;j<M;j++){
      ...
       if(...) break;
       ...
   }
}
```

```
for(int i=0;i<N;i++){
   ...
   for(int j=0;j<M;j++){
      ...
   }
   if(...) break;
}
```

Операторы безусловных переходов

Оператор **goto** – оператор безусловного перехода по метке.

Именем метки может выступать любой идентификатор, не использовавшийся ранее. Метка указывается на новой строке с завершающим двоеточием.

После оператора **goto** указывается имя метки с завершающей точкой с запятой.

Операторы безусловных переходов

Рассмотрим пример: необходимо определить первое трехзначное число сумма крайних цифр которого равна средней цифре. Фрагмент программы осуществляющий это будет иметь вид:

```
  unsigned num = 100;
Next:
  if(num%10 + num/100 == num/10%10) goto Exit;
  num++;
  goto Next;
Exit:
```

Пример 1

Вычислить сумму ряда $\sum_{i=1}^{N}(-1)^i \cdot \frac{2^i}{i!}$ с первой позиции до позиции N с заданной погрешностью. Позиция N и погрешность вводятся пользователем. При реализации оптимизировать вычисления.

```
#include <stdio.h>
int main(int argc, char *argv[])
{
  unsigned n = 0;
  double eps = 0.0;
  printf("Введите позицию: "); scanf("%u",&n);
  printf("Введите погрешность: "); scanf("%lf",&eps);
  double pow2 = 1.0, fact = 1.0, summa = 0.0;
```

Пример 1 (продолжение)

```
  for(unsigned i=1;i<=n;i++){
    pow2 *= 2.0;
    fact *= i;
    double cur = pow2/fact;
    summa += ((i%2==0)?cur:-cur);
    if(cur < eps) break;
  }
  printf("Сумма ряда: %.5lf\n",summa);
  return 0;
}
```

Пример 2

Вывести на экран таблицу значений одной из функций (sin, cos, sqrt, exp) на интервале [A,B]. Таблица состоит из двух колонок (аргумент и функция) и N строк. Значения вывести с точностью до четвертого знака после запятой. Ширина колонок: 15 символов. Значения A, B и N вводит пользователь. Функцию выбирает пользователь.

Пример 2 (продолжение)

```
#include <stdio.h>
#include <math.h>
int main(int argc, char *argv[])
{
  double a,b,step;
  unsigned N,I;
  printf("Введите интервал:\n");
  do{
    scanf("[%lf,%lf]",&a,&b);
    if(a<b) break;
    printf("Некорректный интервал!\n");
  }while(1);
  printf("Введите число строк:\n");
  do{
    scanf("%u",&N);
    if(N>1) break;
    printf("Слишком мало строк!\n");
  }while(1);
```

Пример 2 (продолжение)

```
step = (b - a)/(N - 1);
printf("Выберите функцию:/
        \n 1 - sin\n 2 - cos\n 3 - sqrt\n 4 - exp\n");
do{
  scanf("%u",&I);
  if((I>0)&&(I<5)) break;
  printf("Неправильный номер функции!\n");
}while(1);
printf("----------------------------\n");
printf("    Аргумент   |   Функция     |\n");
printf("----------------------------\n");
double x = a;
```

Пример 2 (продолжение)

```
  while(x<=b){
    double y = 0.0;
    switch(l){
      case 1: {y = sin(x); break;}
      case 2: {y = cos(x); break;}
      case 3: {y = sqrt(x); break;}
      case 4: {y = exp(x); break;}
    }
    printf("   %10.4lf  |   %10.4lf  |\n",x,y);
    x += step;
  }
  printf("-----------------------------\n");
  return 0;
}
```

Пример 3

Определить количество N-значных целых положительных чисел, сумма цифр которых кратна числу K. Числа N и K вводит пользователь. 0<N<10, 1< K<10.

```
#include <stdio.h>
#include <math.h>
```

Пример 3 (продолжение)

```
int main(int argc, char *argv[])
{
  unsigned n = 0, k = 0;
  printf("Введите количество значащих цифр в числах: ");
  do{
    scanf("%u",&n);
    if((n>0)&&(n<10)) break;
    printf("Некорректное значение!\n");
  }while(1);
  printf("Введите число: ");
  do{
    scanf("%u",&k);
    if((k>1)&&(k<10)) break;
    printf("Некорректное значение!\n");
  }while(1);
```

Пример 3 (продолжение)

```
  unsigned lim = (unsigned)pow(10.0,n), count = 0;
  for(unsigned i=lim/10;i<lim;i++){
    unsigned val = i, sum = 0;
    while(val>0){
      sum += val%10;
      val /= 10;
    }
    if(sum % k == 0) count++;
  }
  printf("Количество чисел: %u\n",count);
  return 0;
}
```