Речное божество Нила изображено в виде полулежащего старика с длинной бородой

страны фараонов, а роспись потолка напоминает декоративное оформление заупокойных храмов и гробниц. Знал ли художник Игнатий Игнатьевич Нивинский, делая свои вольные «вариации на тему» в 1912 г., что немногим больше чем через десятилетие сам будет участвовать в росписи другой гробницы, уже новейшего времени? Как бы то ни было, именно он руководил созданием интерьера мавзолея В.И. Ленина в 1924 г. и создал там траурный фриз.

Сейчас цветаевская коллекция слепков, более чем наполовину состоявшая из копий шедевров античности, занимает лишь треть тех залов исторического здания музея, которые его создатель для этого определил. Что-то погибло в ходе пожара 1904 г. еще во время строительства, но большую часть этой коллекции можно увидеть в одном из относительно новых отделов ГМИИ — Учебном художественном музее, носящем имя ученого, в составе Музейного центра Российского государственного гуманитарного университета. Сразу подчеркнем, что вход в него открыт и вышкинцам, и людям, давно вышедшим из студенческого возраста (или еще не вступившим в оный). Сам факт обитания музейных экспонатов в университетских стенах очень показателен — это тот союз, который Иван Владимирович лишь одобрил бы. Ведь именно в музеях то, что в университетах читают и слушают (и не только в рамках тех дисциплин, которые принято называть гуманитарными), обретает объем и краски. Становится тем, что можно потрогать. Но мы с вами, разумеется, не будем делать этого понапрасну, проявляя должное уважение. ✦

Екатерина Потолова

Наталья Ростовцева:
«Наряду с гражданами субъектами гражданского права выступают юридические лица»

Игорь Данилевский:
«Восстановление экономики русских земель после нашествия и возрождение городов как центров ремесла и торговли создали условия для возобновления и укрепления постоянных связей между ними»

МОДУЛЬ ТРЕТИЙ
3 ЗАДАННАЯ ТЕМА

Михаил Валерьянович Курак,
доцент кафедры управления разработкой программного обеспечения
Ирина Николаевна Лесовская,
доцент кафедры архитектуры программных систем

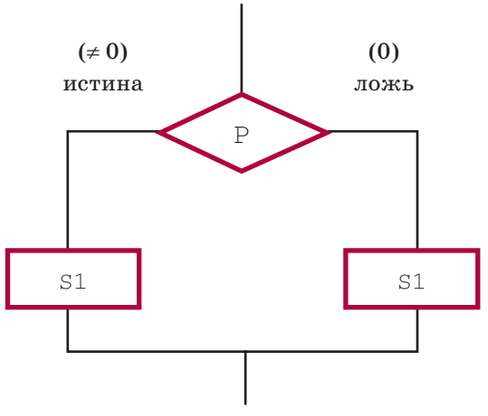
Управляющие конструкции в алгоритмических языках программирования

Языки программирования делятся на две большие группы: алгоритмические (ПАСКАЛЬ, ФОРТРАН, АЛГОЛ, БЕЙСИК, СИ и др.) и неалгоритмические языки (ЛИСП, ПРОЛОГ). Все языки первой группы содержат полный набор управляющих структур, а именно ***следование***, ***ветвление***, ***цикл***.

Следование реализуется в виде последовательности операторов. Остальные конструкции реализуются при помощи специальных операторов, которые принято называть управляющими конструкциями.

Итак, начнем с ветвления. Новичку проще всего представить дерево, от ствола которого идут побеги. А при отсутствии воображения обратимся к... русским народным сказкам. Помните проблему принятия решения всадником на перепутье (направо пойдешь — коня потеряешь, налево пойдешь — меч потеряешь, прямо пойдешь — головы не сносить)? Это и есть модель типичного ветвления «если — то».

Представление управляющей конструкции «ветвление»



`if(p)` — условное выражение

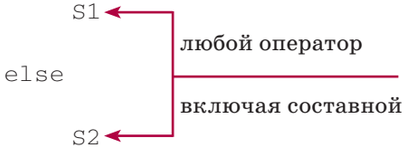


Схема выполнения:

1) Вычисляется выражение `P`.

2) Если P истинно (≠ 0), то выполняется оператор `S1`.

3) Если P ложно (= 0), то выполняется оператор `S2`.

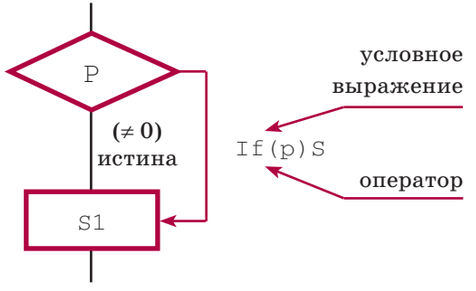
4) Далее выполняется следующий за ветвлением оператор.

Пример:

$$y = \begin{cases} x^2 + 5, \text{x} < 2.1 \\ \sqrt{x-1}, x \geq 2.1 \end{cases}$$

```
if (x < 2.1)
```
$$y = x * x + 5{,}0;$$
```
else
```
$$y = sqrt(x-1);$$

Допускается сокращенное ветвление, когда ветвь `else` отсутствует, то есть имеется только одна ветвь «да» (при истинном значении условного выражения `P`).



Реализуем предыдущий пример на основе сокращенного ветвления

```
y = sqrt (x – 1);
if (x < 2.1) y = x * x + 5.0;
if (num > 6)
```

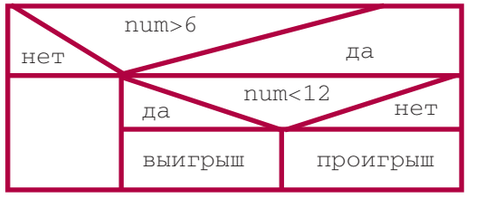
Внимание! *При использовании вложенных* `if`*, если else меньше, чем if, то else принадлежит ближайшему* `if`.

```
if(num < 12)
    printf("выигрыш/n")
else
    printf("проигрыш/n")
```

Сокращ. if

если `num` = 5, то печати нет;
`num` = 10, то выигрыш;
`num` = 15, то проигрыш.

Ниже приведена структурограмма данного алгоритма. Подобная форма записи алгоритмов имеет более компактный вид по сравнению с традиционной.



Представление управляющей конструкции «цикл»

В языке `C` есть три вида управляющих конструкций, реализующих итерационные (повторяющиеся или циклические) процедуры:

– цикл с предусловием;
– цикл с постусловием;
– универсальный цикл.

Цикл с предусловием

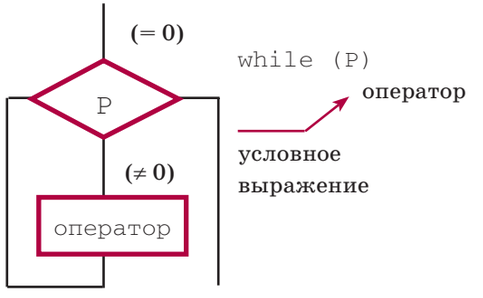


Схема выполнения алгоритма:

1) Вычисляется выражение P.

2) если P истинно , то выполняется оператор цикла, в противном случае цикл завершается.

```
index = 2;
  while (index < 5)
    puts (“плохая погода”);
```

Листинг выполнения программы:

```
плохая погода
плохая погода
```

и так до бесконечности

Почему до бесконечности будет выводиться одна и та же фраза? Причина очевидна: в данном цикле отсутствует точка выхода. Такой цикл называется ***вечным циклом***. Условие выполнение цикла P, которое проверяется перед очередной итерацией, в данном случае не меняется.

Попробуем образовать ***составной оператор*** цикла, заключив в фигурные скобки некую последовательность операторов.

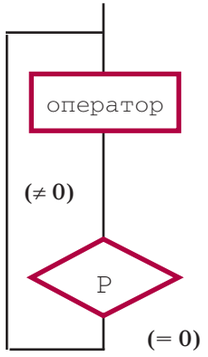
```
index = 2;
while (index < 5)
  {
  puts (“плохая погода”);
  index ++;
  }
```

Листинг выполнения программы:

```
index                    index
2      плохая погода      3
3      плохая погода      4
4      плохая погода      5
          выход
```

Внимание! *Если изначально условие* P *ложно, то тело цикла ни разу не выполняется.*

Цикл с постусловием




```
do
оператор
while (P)
```

Схема выполнения алгоритма:

1) Выполняется оператор цикла.

2) Вычисляется значение выражения P. Если P истинно, то снова выполняется оператор, иначе цикл заканчивается.

```
index = 2
do
 {
  puts (“хорошая погода”);
  index ++
  }
while (index < 5);
```

Листинг выполнения программы:

```
index
хорошая погода
хорошая погода
хорошая погода
выход
```

Даже если задать index = 10, то «хорошая погода» появится на экране один раз, несмотря на то что условие ложное.

Внимание! *Оператор в цикле с постусловием будет выполнен хотя бы один раз при любом значении условия P.*

Универсальный цикл

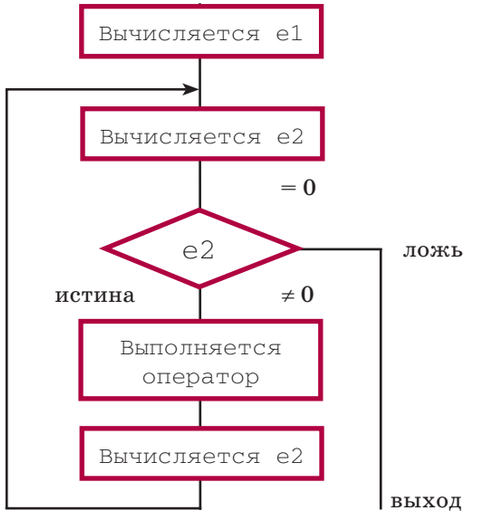
for (e1; e2; e3) оператор,

где e1 — выражение инициализации,

e2 — выражение условия,

e3 — выражение модификации.

Схема выполнения



Отсутствующее значение e2 означает, что значение e2 всегда истинно.

```
for (;;) puts (“бесконечность”);
```

Обратите внимание на приведенный выше пример заголовка универсального цикла, где отсутствуют e1, e2 и e3. Такой цикл приводит к бесконечному выводу на экран фразы «бесконечность», то есть является бесконечным.

Для управления выполнением тела цикла дополнительно используются операторы:

break — принудительное завершение цикла и выход из него;

continue — прекращение очередного шага и переход на следующий шаг (e3 при этом вычисляется).

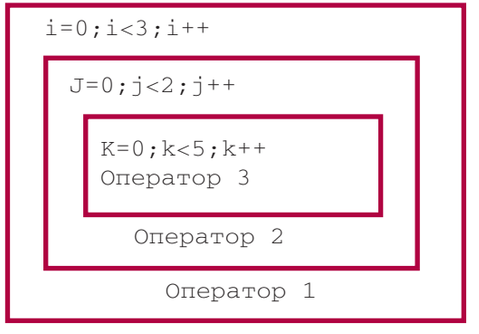
```
for (i = 0; i < 10; i ++)
 {
 puts (“доброе”);
 if (i == 1||i == 2) continue;
 puts (“утро”);
 if (i > 2) break;
 }
```

Листинг выполнения программы:

```
          ii
0 доброе 1
   утро
1 доброе 2
2 доброе 3
3 доброе 4
   утро
```

Во всех конструкциях вместо оператора можно использовать любую из рассмотренных управляющих конструкций.

В частности, внутри цикла можно использовать другой цикл.

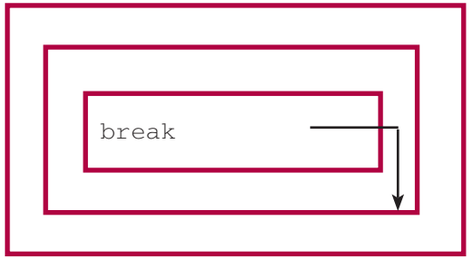


На каждом шаге внешнего цикла вложенный в него цикл будет выполнен столько раз, сколько предписано его условием повтора.

В данной структуре выполняется:

```
оператор 1 ⇒ 3 раза
оператор 2 ⇒ 3 × 2 = 6 раз
оператор 3 ⇒ 3 × 2 × 5 = 30 раз.
```

Для вложенных циклов оператор break действует следующим образом: производится принудительное завершение цикла и выход в предыдущий внешний цикл.

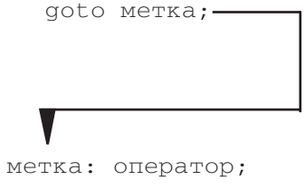



```
for (i = 0; i < 3; i ++)
  {
  puts (“доброе”);
  for (j = 0; j < 2; i ++)
    {
    if (i > 0) break;
    puts (“утро”);
    }
```

Листинг выполнения программы:

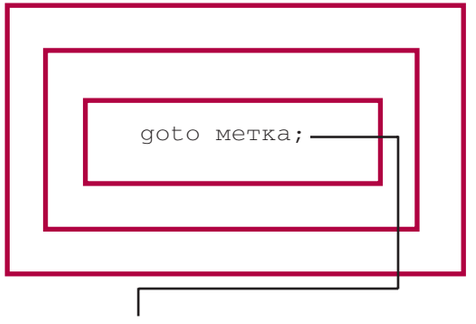
```
I = 0    доброе
j = 0    утро            j = 1
j = 1    утро       j = 2 i = 1
i = 1    доброе
j = 0 → выход на break   i = 2
i = 2    доброе
j = 0 → выход на break   i = 3
```

Принудительный выход **в любую точку программы можно выполнить оператором безусловного перехода с использованием метки.**



Имя метки (идентификатор) составляется по тем же правилам, что и имя любой переменной.

Чаще всего оператор безусловного перехода используется для принудительного выхода из внутреннего цикла за пределы всех внешних циклов.



Пример: проверить тождественность двух логических функций трех переменных.

```
# include <stdio.h>
# include <conio.h>
void main ()
  {
  int a, b, c, y1, y2;
  for (a = 0; a < 2; a ++)
  for (b = 0; b < 2; b ++)
  for (c = 0; c < 2; c ++)
    {
    y1 = a&&b&&c;
    y2 = !a||b&&c;
    if (y1! = y2) goto STOP;
      }
      puts (“Тождественны”);
      goto END;
      STOP:
      puts (“не тождественны”);
      END:
      puts (“проверено!”);
      }
```


Рассмотрим несколько задач по изученной теме.

Задача 1. Разработать программу для проверки натурального четырехзначного десятичного числа и формирования признака проверки по правилу:

1 — счастливое число;

0 — несчастливое число.

Счастливым считается число, у которого сумма двух младших цифр равна сумме двух старших цифр. Например: 2341 — счастливое число, а 3145 — нет.

```
# include <stdio.h>
void main ()
  {
intch, // исходное число
copia, // копия исходного числа
cifra, // очередная цифра числа
sh, // сумма двух старших цифр
sl, // сумма двух младших цифр
flag; // признак счастливого числа
printf (“Введите число:”); scanf
(“%d”, &ch);
copia = ch;
sh = sl = 0;
cifra = copia % 10; sl += cifra;
copia /= 10;
cifra = copia % 10; sl += cifra;
copia /= 10;
cifra = copia % 10; sh += cifra;
copia /= 10;
cifra = copia % 10; sh += cifra;
copia /= 10;
flag = sl == sh;
printf (“Признак для числа %d
равен %d\n”, ch, flag);
  }
```

Данный алгоритм реализуется при использовании следования. Итерационные процедуры (циклы) не использованы, поэтому программа имеет ограничение по разрядности исходного числа. В задаче используется операция «остаток от деления» (обозначается как %, используется только для целых чисел), что позволяет «отрезать» последнюю цифру от копии исходного числа. Далее число последовательно делится на 10, уменьшаясь на один разряд.

Задача 2. Определите результат, выведенный на экран приведенной программой, если с клавиатуры введены числа 2.25 и 3.75.

```
# include <stdio.h>
void main ()
  {
float x1, x2, s, sr;
int    cx1, cx2;
printf (“Введите два веществен-
ных положительных числа:”);
scanf (“%f%f”, &x1, &x2);
s = (x1 – (int) x1) + (x2 –
(int) x2); // сумма целых ча-
стей
cx1 = x1;// cx1 присваивается
целая часть x1
cx2 = x2;
sr = (float) (cx1 + cx2) / 2;
printf (“%.3f %.3f”, s, sr);
  }
```

Вывод:

```
1) 6.000  3.000  2) 1.000  2.500
3) 5.000  0.500  4) 2.500  1.000.
```

Следует заметить, что процедура явного приведения к вещественному типу данных суммы переменных cx1 и cx2 позволяет избежать потери значимости. Результат будет вещественным. В противном случае результатом целочисленного деления было бы целое число, то есть результат выполнения операции (2+3)/2 был бы равен 2.

Задача 3. Определите результат, выведенный на экран приведенной программой, если с клавиатуры введено число 253.

```
  void main ()
  {
intch, copia, c, n;
printf («Введите число:»);
scanf («%d», &ch);
copia = ch;
n = 0;
c = copia % 10; n = n * 10 + c;
copia /= 10;
c = copia % 10; n = n * 10 + c;
copia /= 10;
c = copia % 10; n = n * 10 + c;
copia /= 10;
printf («%d», n);
  }
```

Вывод:

1) 253 2) 532
3) 352 4) 10.

Задача 4. Разработать программу для вычисления значения функции G = F (X, Y)

| 'Y', если точка с координатами (X, Y) попадает в фигуру,
| G = <
| 'Y', если точка с координатами (X, Y) не попадает в фигуру.

Фигура — сектор круга радиусом R = 2 в диапазоне углов 270 <= fi <= 45.

```
# include <stdio.h>
# define R 2.0
void main ()
  {
floatx, y; // Координаты точки
int g; // Значение функции
printf  (“Введите  координаты
точки:”);  scanf  (“%f%f”,  &x,
&y);
if (x * x + y * y <= R * R) //
// в круге
if (x >= 0) // справа от оси Y
      if (y <= x) //ниже прямой
y = x, следовательно, в фигуре
      g = 'Y';
      else
      g = 'N'; // выше прямой
// y = x
else
      g = 'N'; // слева от оси
// Y
else
      g = 'N'; // вне круга
printf    («G(%.2f,%.2f)=%c\n»,
x, y, g);
  }
```

Данная реализация предполагает вложенность ветвлений, где изначально проверяется условие попадания в круг.

Задача 5. Вывести на экран все числа Фибоначчи, не превышающие заданного значения и являющиеся простыми числами.

```
# include <stdio.h>
void main ()
  {
Intfib, // Очередное число Фи-
// боначчи
limit, // Предельное значение
// для перебора чисел
f1, // Предыдущее число Фибо-
// наччи (i — 1oe)
f2, //Предыдущее число Фибо-
// наччи (i — 2oe)
del, // Очередной делитель при
// проверке на простое число
k, // Количество простых чисел
// среди чисел Фибоначчи
flag; // Индикатор простого
//числа
printf  (“Введите  предельное
значение:”);
scanf (“%d”, &limit);
f1 = 0;
f2 = 1;
fib = 1;
k = 0;
while (1) // Цикл вычисления чисел
// Фибоначчи
  {
if (fib > limit) break;
for (flag = 1, del = 2; del <
fib; del ++) // Цикл проверки
// на простое
if (fib % del == 0)
  {
flag = 0;
break;
  }
  if  (flag  ==  1)  //Анализ
// результата цикла проверки на простое
      {
      printf (“%5d”, fib);
      k ++;
      }
      fib = f1 + f2;
      f1 = f2;
      f2 = fib;
      } // конец цикла вычисления
// чисел Фибоначчи
printf  (“\n  Таких  чисел  %d
шт.\n”, k);
  }
```

Задача 6. Определите результат, выведенный на экран приведенной программой, если с клавиатуры введены числа ch = 456771 и k = 2 45

```
*/
# include <stdio.h>
void main ()
{
intch, copia, m, d; n, k, ost,
i;
printf    (“Введите    значение
ch:”);
scanf (“%ld”, &ch);
printf (“Введите значение k:”);
scanf (“%d”, &k); copia = ch;
n = 0;
while (copia ! = 0)
  {
     N ++;
     copia /= 10;
  }

for (ost = n — k, d = 1, I = 0;
I < ost; I ++) d *= 10;
m = ch / d;
printf (“%d”, m);
  }
```

Попробуйте определить результат работы программы самостоятельно. В случае необходимости программу необходимо реализовать на ЭВМ.

Практика показывает, что с циклами (в том числе и с вложенными) ученики осваиваются в полной мере при работе с массивами. Этой теме будет посвящено следующее занятие. ✦