Урок 6. Функции

По мере дальнейшего прохождения курса программы будут становиться всё более интересными, но и одновременно более объёмными и сложными. Для облегчения создания больших программ в языке Си существуют функции, которые позволяют разбивать программу на несколько независимых частей, пригодных для многократного использования.

Вы уже сталкивались с функциями. Выполнение программы начинается с функции `main`, для ввода-вывода используются функции `printf`, `scanf` и др.

Общий вид функции

```
тип_результата имя_функции (объявления аргументов)
{
    объявления и инструкции
}
```

Для возврата из функции используется оператор `return`:

```
return выражение;
```

Функция возвращает значение `выражения`, указанное после ключевого слова `return`. В качестве типа результата может быть указан `void`, это означает, что функция не возвращает никаких значений (см. примеры функций); при этом оператор `return` записывается как `return;` (без операнда).

Примеры функций

```
/* Возвращает квадрат целого числа */
int sqr(int x)
{
  int y;
  y = x*x;
  return y;
}
/* Нахождение среднего арифметического
   вещественных чисел */
int average(int a, int b)
{
  int r;
  r = (a * b) / 2.0;
  return r;
}
/* Вывод целого числа на экран */
void printint(int num)
{
  printf("%d ", num);
}
```

ВНИМАНИЕ! ***Любая функция должна быть определена до того, как её вызовут (т.е. «выше» по тексту программы, чем её вызов).***

Задание 1

Написать программу, которая выводит таблицу кубов чисел от 1 до 10. В ней обязательно должна быть функция, вычисляющая куб целого числа.

Решение

```
#include <stdio.h>
#include <stdlib.h>
/* Функция вычисляет куб числа */
int cube(int x)
{
  int y;
  y = x*x*x;
  return y;
}
/* С этой функции начинается выполнение программы */
void main()
{ int i;
  for (i = 1; i <= 10; i++)
    printf("%d %d\n", i, cube(i));
  system("PAUSE");
}
```

Пояснения к тексту программы

`int cube(int x)`

Объявление функции с именем `cube`. У этой функции единственный аргумент `x` типа `int`. Она возвращает значение типа `int`. **Тело функции** заключено в операторные скобки `{ }`.

```
int y;
```

Объявление **локальной переменной** `y` типа `int`. Она «видна» только в самой функции `cube`, и нигде более. Ранее Вы уже использовали локальные переменные в функции `main`.

```
y = x*x*x;
```

Вычисление куба аргумента функции – переменной `x`.

```
return y;
```

Оператор `return` прекращает выполнение функции, значение переменной y возвращается функцией `cube`.

```
for (i = 1; i <= 10; i++)
      printf("%d %d\n", i, cube(i));
```

В этом цикле происходит расчёт и вывод таблицы кубов целых чисел от 1 до 10. Выражение `cube(i)` – это вызов функции `cube`; в качестве аргумента передаётся значение переменной `i`. Функция возвратит куб значения `i`, который и будет выведен.

Задание 2

Написать программу для расчёта периметра прямоугольника с помощью функции.

Решение

```
#include <stdio.h>
#include <stdlib.h>
/* Функция вычисляет периметр прямоугольника */
double perim(double a, double b)
{
  int p;
  p = 2.0*(a+b);
  return p;
}
/* Начало программы */
void main()
{ // Инициализация переменных
  double a, b;
  // Ввод a и b
  printf("Вычисление периметра прямоугольника\n");
  printf("Введите a: ");
  scanf("%lf",&a);
  printf("Введите b: ");
  scanf("%lf",&b);
  // Вычисление периметра и вывод результата
  printf("P = %lf", perim(a, b));
  // Ожидание нажатия клавиши
  system("PAUSE");
}
```

Пояснения к тексту программы

`double perim(double a, double b)`

Функция `perim` вычисляет и возвращает периметр прямоугольника (тип `double` – вещественное число). У этой функции два аргумента – длины сторон прямоугольника, переменные `a` и `b` типа `double`.

`printf("P = %lf", perim(a, b));`

Расчёт периметра прямоугольника с помощью вызова функции `perim`. Результат передаётся функции `printf`, которая и выводит его на экран.

Задание 3

Напишите программу, выводящую 10 целых случайных чисел в диапазоне от 1 до n (n вводится с клавиатуры). В ней должны быть три функции

- `void randomize()` – сброс генератора случайных чисел
- `int  random(int n)` – возвращает случайное число от 1 до n
- `void main()` – основная программа

Решение

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

/*** Функции ***/
/* Сброс генератора случайных чисел */
void randomize()
{
  srand(time(NULL));
}
/* Генерация случайного числа от 1 до n */
int  random(int n)
{
  return 1 + rand()*n /(RAND_MAX+1);
}

/*** Главная функция ***/
void main()
{
  int i, n;
  // Ввести диапазон
  printf("Вывод случайных чисел от 1 до n\n");
  printf("Введите n: ");
  scanf("%d", &n);
  // Сбросить генератор случайных чисел
  randomize();
  // Вывести десять случайных чисел
  for (i = 0; i < 20; i++)
    printf("%d\n", random(n));
  // Ожидать нажатия клавиши
  while(!kbhit());
}
```

Задание 4

Напишите программу, которая выводит цветные звёздочки на экран. Для вывода отдельной звёздочки напишите функцию

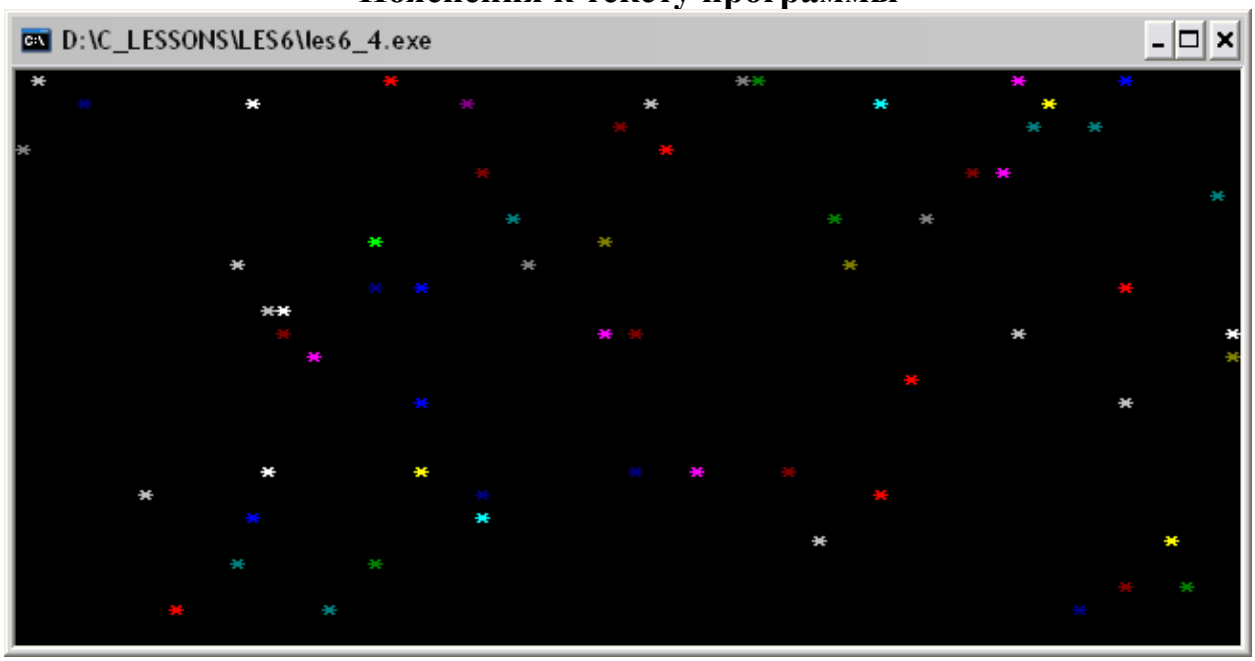
`void star(int x, int y, int c).`

Программа должна завершаться при нажатии клавиши. Для вывода цветного текста и управления курсором используйте библиотеку XCRT (см. сайт курсов). Пример вывода программы – см. в пояснениях к тексту программы

Решение

```
#include <crt.h>    // Библиотека расширенного ввода-вывода
#include <stdlib.h> // Библиотека общего назначения
#include <time.h>   // Библиотека работы со временем
/* Функция выводит звёздочку
   в столбце x, строке y цвета c */
void star(int x, int y, int c)
{
  gotoxy(x, y); // Переместить курсор в нужное место
  textcolor(c); // Установить цвет текста
  cprintf("*"); // Вывести звёздочку
}
/* Начало программы */
void main()
{ // Объявление переменных
  int x, y, c;
  // Подготовка окна программы
  setsize(80, 25); // Установить размер окна 80x25 символов
  textbackground(BLACK); // Цвет фона - чёрный
  clrscr(); // Очистить окно цветом фона
  // Сброс генератора случайных чисел
  srand(time(NULL));  // time(NULL) - текущее время в
секундах
  // Цикл вывода звёздочек
  while(!kbhit()) // Условие -- "пока не нажата клавиша"
  { // Генерация случайных чисел для функции star
    x = 1 + 80*rand() / (RAND_MAX+1); // Номер столбца
    y = 1 + 25*rand() / (RAND_MAX+1); // Номер строки
    c = 1 + 15*rand() / (RAND_MAX+1); // Цвет (от 1 до 15)
    star(x, y, c); // Нарисовать звёздочку
    delay(500);    // Задержка в 0.5 мс
  }
}
```

Пояснения к тексту программы




```
void star(int x, int y, int c)
```

Эта функция выводит на экран цветную звёздочку. Аргументы `x` и `y` задают координаты звёздочки (`x` – столбец, `y` – строку), а `c` – цвет (значение от 0 до 15; расшифровка номеров цветов есть в документации к библиотеке XCRT и файле `crt.h`).

```
gotoxy(x, y); // Переместить курсор в нужное место
```

Вызов функции gotoxy, перемещающей курсор в столбец x и строку y.

```
textcolor(c); // Установить цвет текста
```

Установка текущего цвета текста. Именно этим цветом будет выводиться звёздочка.

```
cprintf("*"); // Вывести звёздочку
```

Вывод звёздочки на экран. Функция `cprintf` – это вариант `printf`, который выводит текст с учётом текущего цвета текста и текущей позиции курсора. В отличие от `printf`, он не задерживает вывод, что важно для программирования анимации.

```
x = 1 + 80*rand() / (RAND_MAX+1); // Номер столбца
y = 1 + 25*rand() / (RAND_MAX+1); // Номер строки
c = 1 + 15*rand() / (RAND_MAX+1); // Цвет (от 1 до 15)
```

В этих строках задаются случайным образом координаты и цвет звёздочки. Т.к. диапазон функции `rand()` не регулируется, приходится использовать довольно сложные выражения. Значения x получаются в диапазоне от 1 до 80, для y – от 1 до 25, для c – от 1 до 15.

Задание 5

Напишите программу, которая показывает «мультик» - движущуюся взад-вперёд «картинку» из символов (или строку текста). Используйте функции библиотеки XCRT (можно взять на сайте курсов).

Решение

```
#include <crt.h>
/* Показать строку текста */
void showobj(int x, int y)
{
  gotoxy(x, y);
  textcolor(YELLOW);
  cprintf("Hello, ");
  textcolor(LIGHTBLUE);
  cprintf("World");
}
/* Стереть строку текста */
void hideobj(int x, int y)
{
  gotoxy(x, y);
  cprintf("            ");
}
/* Начало программы */
void main()
{ int i;
  // Инициализация окна
  setsize(80, 25); // Установить размер окна
  clrscr();        // и очистить его
  // Основной цикл
  while (1)
  { // Движение строки вправо
    for (i = 1; i < 80 - 12; i++)
    {
      showobj(i, 12); // Показать строку
      delay(100);     // Задержка 100 мс
      hideobj(i, 12); // Скрыть строку
    }
    // Движение строки влево
    for (i = 80 - 12; i >= 1; i--)
    { showobj(i, 12); // Показать строку
      delay(100);     // Задержка 100 мс
      hideobj(i, 12); // Скрыть строку
    }
  }
}
```

Пояснения к тексту программы

```
void showobj(int x, int y)
```

Эта функция выводит на экран строку текста «Hello, World». Слово Hello печатается жёлтым цветом, а World – голубым. Этой функции требуется два аргумента типа `int` – координаты начала строки `x` и `y`.

```
gotoxy(x, y);
```

Перевод курсора на столбец x и строку y. Именно с этой позиции будет выведена строка текста.

```
textcolor(YELLOW);
cprintf("Hello, ");
```

Функция textcolor устанавливает текущий цвет текста (в данном случае жёлтый), а cprintf выводит слово «Hello, ».

```
textcolor(LIGHTBLUE);
cprintf("World");
```

Вывод слова «World» синим цветом. Оно будет выведено сразу за словом «Hello».

```
void hideobj(int x, int y)
```

Функция стирает строку текста «Hello, World». Аргументы аналогичны тем, что есть у функции `showobj`.

```
while (1)
```

Это – бесконечный цикл со всегда истинным условием. Условие `(1)` – это сокращённая запись `(1 != 0)`. Поэтому программа не реагирует на нажатия клавиш, но её можно остановить, просто закрыв окно. Здесь это сделано для упрощения программы, на следующих уроках Вы узнаете, как выйти из такого цикла.

```
for (i = 1; i < 80 - 12; i++)
```

В этом цикле обеспечивается движение строки слева направо. Сначала строка печатается без отступа, потом с отступом один столбец от края и т.д. до конца.

```
showobj(i, 12); // Показать строку
delay(100);     // Задержка 100 мс
hideobj(i, 12); // Скрыть строку
```

Именно эта часть программы обеспечивает анимацию. Сначала функция `showobj` показывает строку текста, затем функция `delay` останавливает выполнение программы на 0,1 с. и глаз успевает заметить строку. После этого функция `hideobj` стирает её.

При следующем выполнении тела цикла переменная `i` будет иметь другое значение и строка появится несколько правее. Именно это создаёт иллюзию движения.

Задание 6*

Напишите программу, которая будет показывать «летающий» по всему окну мячик (примерно так, как в играх типа «Арканоид»).

Решение

```
#include <crt.h>

void main()
{ // Инициализация переменных
  const int xsize = 80;// Ширина окна
  const int ysize = 50;// Высота окна
  int x = xsize / 2;   // Координата x мячика
  int y = ysize / 2;   // Координата y мячика
  int dx = 1, dy = -1; // Направление движения мяча
  // Инициализация окна
  setsize(xsize, ysize); // Установить размер окна
  settitle("BALL");      // Установить заголовок окна
  textcolor(YELLOW);     // Цвет мяча - жёлтый
  clrscr();              // Очистить экран
  hidecursor();          // Спрятать курсор
  // Основной цикл программы
  while(!kbhit()) // Условие - "пока не нажата клавиша"
  { // 1. Стереть старое изображение мяча
    gotoxy(x, y);
    cprintf(" ");
    // 2. Расчёт новых координат мяча от стенки
    // 2.1. Отражение от "стенок" - границ окна
    if      (y == 1     && dy == -1) dy = 1;
    else if (y == ysize && dy ==  1) dy = -1;
    if      (x == 1     && dx == -1) dx = 1;
    else if (x == xsize && dx ==  1) dx = -1;
    // 2.2. Изменение координат
    x += dx;
    y += dy;
    // 3. Показать мяч
    gotoxy(x, y);
    cprintf("O");
    delay(50);    // Задержка 50 мс
  }
}
```

Пояснения к тексту программы

```
int dx = 1, dy = -1; // Направление движения мяча
```

В этих двух переменных задано направление движения мяча. При каждом его смещении к координате x прибавляется переменная `dx`, а к координате `y` – переменная `dy`. Такой механизм позволяет легко менять направление движения при отражения от стенок.

```
while(!kbhit())
```

Этот цикл `while` будет выполняться до тех пор, пока не нажата клавиша. Напоминаем, что `(!kbhit())` – это сокращение от (`kbhit() != 0`). Это условие проверяется каждый раз перед выполнением тела цикла, поэтому из программы можно всегда выйти, нажав любую клавишу.

```
if      (y == 1     && dy == -1) dy = 1;
else if (y == ysize && dy ==  1) dy = -1;
```

Здесь запрограммировано отражение мяча от верхней и нижней границ окна. Оператор `&&` - это логическое «И» (условие будет истинно только в том случае, если оба разделённые && условия истинны, т.е. выполнены). Напоминаем, что есть также оператор `||` - логическое «ИЛИ» и оператор `!` – логическое «НЕ».

В первом условии проверяется, пересекает ли мяч верхнюю границу окна, и если да – то он «отскакивает» от неё (направление движения по вертикали меняется на противоположное).

Во втором условии проверяется, пересекает ли мяч нижнюю границу окна.

```
if      (x == 1     && dx == -1) dx = 1;
else if (x == xsize && dx ==  1) dx = -1;
```

Здесь происходит «отскок» мяча от левой и правой границ окна. Логика здесь совершенно аналогична таковой для верхней и нижней границ.

```
x += dx;
y += dy;
```

Изменение координат мяча согласно направлению его движения

Резюме

Общий вид функции

```
тип_результата имя_функции (объявления аргументов)
{
    объявления и инструкции
}
```

Для возврата из функции используется оператор `return`:

```
return выражение;
```

Тип `void` – функция не возвращает никаких значений. В этом случае пишут `return` без операнда (а если он нужен в самом конце – можно его и не писать):

```
return;
```

Примеры функций

```
/* Возвращает квадрат целого числа */
int sqr(int x)
{
  int y;
  y = x*x;
  return y;
}
/* Нахождение среднего арифметического
    вещественных чисел */
int average(int a, int b)
{
  int r;
  r = (a * b) / 2.0;
  return r;
}
/* Вывод целого числа на экран */
void printint(int num)
{
  printf("%d ", num);
}
```

ВНИМАНИЕ! ***Любая функция должна быть определена до того, как её вызовут (т.е. «выше» по тексту программы, чем её вызов).***

Упражнения

1. Напишите функцию `double hyp(double x)`, которая вычисляет выражение 1/x, и выведите её значения для x в интервале [0,1;2] с шагом 0,1.
2. Напишите функцию, которая вычисляет периметр треугольника по трём сторонам и испытайте её.
3. Переделайте программу из **задания 4** таким образом, чтобы функция `star` не требовала аргументов и имела вид `void star()`.
4. Доработайте **упражнение 3** таким образом, чтобы вместо звёздочек программа печатала случайные символы с кодами от 32 до 255.

Упражнения повышенной сложности

5. Реализуйте функцию `int round(double x)`, которая округляет число по правилам математики и испытайте её. Код вида:
   ```
   int x;
   double y = 1.5;
   x = (int) y;
   ```
   не округляет, а «обрезает» дробную часть значения переменной `y`.
6. Напишите и испытайте функцию, которая рассчитывает площадь треугольника по формуле Герона (нужны три стороны треугольника): $S = \sqrt{p(p-a)(p-b)(p-c)}$, где $p = \frac{a+b+c}{2}$ (полупериметр треугольника).

Творческие упражнения

7. На основе **задания 6** напишите игру «теннис». На экране находятся ракетка и мячик. Мячик «летает» по экрану и отскакивает от ракетки и от верхней и боковых границ окна. Ракетка управляется с клавиатуры (например, клавишами 4 и 6 на цифровой клавиатуре при включённом Num Lock). Цель игры – не дать мячику «упасть». Используйте функции `kbhit` и `getch` из библиотеки `crt.h`.
8. Сделайте программу-«мультфильм» с движущимися кусками цветного текста. Сюжет и сложность – на Ваше усмотрение (реклама, цветные рамки, хранители экрана – «скринсейверы» и многое другое).
9. Добавьте звуковые эффекты в **задание 6** и/или игру «Теннис». Всё необходимое, кроме самих звуковых файлов, есть в библиотеке XCRT.