Решения избранных упражнений по программированию

Упражнения по программированию из главы 2

2.1.

```
/* Упражнение по программированию 2.1 */
#include <stdio.h>
int main(void)
{
   printf("Иван Иванов\n");
   printf("Иван\nИванов\n");
   printf("Иван ");
   printf("Иванов\n");
   return 0;
}
```

2.3.

```
/* Упражнение по программированию 2.3 */
#include <stdio.h>
int main(void)
{
   int ageyears;      /* возраст в годах */
   int agedays;       /* возраст в днях  */
                      /* больший возраст может требовать типа long */
   ageyears = 101;
   agedays = 365 * ageyears;
   printf("Возраст %d лет составляет %d дней.\n", ageyears, agedays);
   return 0;
}
```

2.4.

```
/* Упражнение по программированию 2.4 */
#include <stdio.h>
void jolly(void);
void deny(void);
int main(void)
{
   jolly();
   jolly();
   jolly();
   deny();
   return 0;
}
void jolly(void)
{
   printf("Он веселый молодец!\n");
}
void deny(void)
{
   printf("Никто не может это отрицать!\n");
}
```

2.6.

```
/* Упражнение по программированию 2.6 */
#include <stdio.h>
int main(void)
{
   int toes;
   toes = 10;
   printf("Значение toes = %d\n", toes);
   printf("Удвоенное значение toes = %d\n", 2 * toes);
   printf("Учетверенное значение toes = %d\n", toes * toes);
   return 0;
}
/* или создайте еще две переменные и установите их в 2 * toes и toes * toes */
```

2.8.

```
/* Упражнение по программированию 2.8 */
#include <stdio.h>
void one_three(void);
void two(void);
int main(void)
{
   printf("начинаем:\n");
   one_three();
   printf("порядок!\n");
   return 0;
}

void one_three(void)
{
   printf("один\n");
   two();
   printf("три\n");
}

void two(void)
{
   printf("два\n");
}
```

Упражнения по программированию из главы 3

3.2.

```
/* Упражнение по программированию 3.2 */
#include <stdio.h>
int main(void)
{
   int ascii;
   printf("Введите ASCII-код: ");
   scanf("%d", &ascii);
   printf("%d - это ASCII-код для %c.\n", ascii, ascii);
   return 0;
}
```

3.4.

```
/* Упражнение по программированию 3.4 */
#include <stdio.h>
```

```
int main(void)
{
    float num;
    printf("Введите значение с плавающей запятой: ");
    scanf("%f", &num);
    printf("Запись с фиксированной запятой: %f\n", num);
    printf("Экспоненциальная форма записи: %e\n", num);
    printf("Двоично-экспоненциальное представление: %a\n", num);
    return 0;
}
```

3.6.

```
/* Упражнение по программированию 3.6 */
#include <stdio.h>
int main(void)
{
    float mass_mol = 3.0e-23;    /* масса молекулы воды в граммах */
    float mass_qt = 950;         /* масса кварты воды в граммах   */
    float quarts;
    float molecules;
    printf("Введите количество кварт воды: ");
    scanf("%f", &quarts);
    molecules = quarts * mass_qt / mass_mol;
    printf("%f кварт воды содержит %e молекул.\n", quarts, molecules);
    return 0;
}
```

Упражнения по программированию из главы 4

4.1.

```
/* Упражнение по программированию 4.1 */
#include <stdio.h>
int main(void)
{
    char fname[40];
    char lname[40];
    printf("Введите свое имя: ");
    scanf("%s", fname);
    printf("Введите свою фамилию: ");
    scanf("%s", lname);
    printf("%s, %s\n", lname, fname);
    return 0;
}
```

4.4.

```
/* Упражнение по программированию 4.4 */
#include <stdio.h>
int main(void)
{
    float height;
    char name[40];
    printf("Введите свой рост в дюймах: ");
    scanf("%f", &height);
    printf("Введите свое имя: ");
    scanf("%s", name);
    printf("%s, ваш рост составляет %.3f футов\n", name, height / 12.0);
    return 0;
}
```

4.7.

```
/* Упражнение по программированию 4.7 */
#include <stdio.h>
#include <float.h>
int main(void)
{
   float ot_f = 1.0 / 3.0;
   double ot_d = 1.0 / 3.0;
   printf(" значения float: ");
   printf("%.4f %.12f %.16f\n", ot_f, ot_f, ot_f);
   printf("значения double: ");
   printf("%.4f %.12f %.16f\n", ot_d, ot_d, ot_d);
   printf("FLT_DIG: %d\n", FLT_DIG);
   printf("DBL_DIG: %d\n", DBL_DIG);
   return 0;
}
```

Упражнения по программированию из главы 5

5.1.

```
/* Упражнение по программированию 5.1 */
#include <stdio.h>
int main(void)
{
   const int minperhour = 60;
   int minutes, hours, mins;
   printf("Введите количество минут для преобразования: ");
   scanf("%d", &minutes);
   while (minutes > 0 )
   {
      hours = minutes / minperhour;
      mins = minutes % minperhour;
      printf("%d минут = %d часов, %d минут\n", minutes, hours, mins);
      printf("Введите следующее количество минут (0 для завершения): ");
      scanf("%d", &minutes);
   }
   printf("Программа завершена.\n");
   return 0;
}
```

5.3.

```
/* Упражнение по программированию 5.3 */
#include <stdio.h>
int main(void)
{
   const int daysperweek = 7;
   int days, weeks, day_rem;
   printf("Введите количество дней: ");
   scanf("%d", &days);
   while (days > 0)
   {
      weeks = days / daysperweek;
      day_rem = days % daysperweek;
      printf("%d дней составляют %d недели и %d дней.\n",
             days, weeks, day_rem);
```

```
        printf("Введите следующее количество дней (0 или меньшее значение для
завершения): ");
        scanf("%d", &days);
    }
    printf("Программа завершена.\n");
    return 0;
}
```

5.5.

```
/* Упражнение по программированию 5.5 */
#include <stdio.h>
int main(void) /* finds sum of first n integers */
{
    int count, sum;
    int n;
    printf("Введите верхний предел: ");
    scanf("%d", &n);
    count = 0;
    sum = 0;
    while (count++ < n)
        sum = sum + count;
    printf("sum = %d\n", sum);
    return 0;
}
```

5.7.

```
/* Упражнение по программированию 5.7 */
#include <stdio.h>
void showCube(double x);
int main(void) /* вычисляет куб введенного числа */
{
    double val;
    printf("Введите значение с плавающей запятой: ");
    scanf("%lf", &val);
    showCube(val);
    return 0;
}

void showCube(double x)
{
    printf("Куб %e равен %e.\n", x, x*x*x);
}
```

Упражнения по программированию из главы 6

6.1.

```
/* pe6-1.c */
/* В этой реализации предполагается, что коды символов */
/* являются последовательными, как в кодировке ASCII. */
#include <stdio.h>
#define SIZE 26
int main( void )
{
    char lcase[SIZE];
    int i;
    for (i = 0; i < SIZE; i++)
        lcase[i] = 'a' + i;
```

```
    for (i = 0; i < SIZE; i++)
       printf("%c", lcase[i]);
    printf("\n");
    return 0;
}
```

6.3.

```
/* pe6-3.c */
/* В этой реализации предполагается, что коды символов */
/* являются последовательными, как в кодировке ASCII. */
#include <stdio.h>
int main( void )
{
    char let = 'F';
    char start;
    char end;
    for (end = let; end >= 'A'; end--)
    {
       for (start = let; start >= end; start--)
          printf("%c", start);
       printf("\n");
    }
    return 0;
}
```

6.6.

```
/* pe6-6.c */
#include <stdio.h>
int main( void )
{
    int lower, upper, index;
    int square, cube;
    printf("Введите начальное целое число: ");
    scanf("%d", &lower);
    printf("Введите конечное целое число: ");
    scanf("%d", &upper);
    printf("%5s %10s %15s\n", "число", "квадрат", "куб");
    for (index = lower; index <= upper; index++)
    {
       square = index * index;
       cube = index * square;
       printf("%5d %10d %15d\n", index, square, cube);
    }
    return 0;
}
```

6.8.

```
/* pe6-8.c */
#include <stdio.h>
int main( void )
{
    double n, m;
    double res;
    printf("Введите пару чисел: ");
    while (scanf("%lf %lf", &n, &m) == 2)
    {
       res = (n - m) / (n * m);
```

```
        printf("(%.3g - %.3g)/(%.3g*%.3g) = %.5g\n", n, m, n, m, res);
        printf("Введите следующую пару (или нечисловое значение для завершения): ");
    }
    return 0;
}
```

6.11.

```
/* pe6-11.c */
#include <stdio.h>
#define SIZE 8
int main( void )
{
    int vals[SIZE];
    int i;
    printf("Введите %d целых чисел.\n", SIZE);
    for (i = 0; i < SIZE; i++)
        scanf("%d", &vals[i]);
    printf("Введенные вами значения в обратном порядке:\n");
    for (i = SIZE - 1; i >= 0; i--)
        printf("%d ", vals[i]);
    printf("\n");
    return 0;
}
```

6.13.

```
/* pe6-13.c */
/* Эта версия начинает со степени 0. */
#include <stdio.h>
#define SIZE 8
int main( void )
{
    int twopows[SIZE];
    int i;
    int value = 1; /* 2 в степени 0 */
    for (i = 0; i < SIZE; i++)
    {
        twopows[i] = value;
        value *= 2;
    }
    i = 0;
    do
    {
        printf("%d ", twopows[i]);
        i++;
    } while (i < SIZE);
    printf("\n");
    return 0;
}
```

6.14.

```
/* pe-14.c */
/* Упражнение по программированию 6.14 */
#include <stdio.h>
#define SIZE 8
int main(void)
{
    double arr[SIZE];
```

```
    double arr_cumul[SIZE];
    int i;
    printf("Введите %d чисел:\n", SIZE);
    for (i = 0; i < SIZE; i++)
    {
       printf("значение #%d: ", i + 1);
       scanf("%lf", &arr[i]);
       /* или scanf("%lf", arr + i); */
    }
    arr_cumul[0] = arr[0]; /* установка первого элемента */
    for (i = 1; i < SIZE; i++)
       arr_cumul[i] = arr_cumul[i-1] + arr[i];
    for (i = 0; i < SIZE; i++)
       printf("%8g ", arr[i]);
    printf("\n");
    for (i = 0; i < SIZE; i++)
       printf("%8g ", arr_cumul[i]);
    printf("\n");
    return 0;
}
```

6.16.

```
/* pe6-16.c */
#include <stdio.h>
#define RATE_SIMP 0.10
#define RATE_COMP 0.05
#define INIT_AMT 100.0
int main( void )
{
   double daphne = INIT_AMT;
   double deidre = INIT_AMT;
   int years = 0;
   while (deidre <= daphne)
   {
      daphne += RATE_SIMP * INIT_AMT;
      deidre += RATE_COMP * deidre;
      ++years;
   }
   printf("Размеры вкладов спустя %d лет:\n", years);
   printf("Дафна: $%.2f\n", daphne);
   printf("Дейдра: $%.2f\n", deidre);
   return 0;
}
```

Упражнения по программированию из главы 7

7.1.

```
/* Упражнение по программированию 7.1 */
#include <stdio.h>
int main(void)
{
   char ch;
   int sp_ct = 0;
   int nl_ct = 0;
   int other = 0;
   while ((ch = getchar()) != '#')
   {
```

```
        if (ch == ' ')
            sp_ct++;
        else if (ch == '\n')
            nl_ct++;
        else
            other++;
    }
    printf("количество пробелов: %d, количество символов новой строки: %d,
количество всех остальных символов: %d\n",
            sp_ct, nl_ct, other);
    return 0;
}
```

7.3.

```
/* Упражнение по программированию 7.3 */
#include <stdio.h>
int main(void)
{
    int n;
    double sumeven = 0.0;
    int ct_even = 0;
    double sumodd = 0.0;
    int ct_odd = 0;
    while (scanf("%d", &n) == 1 && n != 0)
    {
        if (n % 2 == 0)
        {
            sumeven += n;
            ++ct_even;
        }
        else // n % 2 равно 1 или -1
        {
            sumodd += n;
            ++ct_odd;
        }
    }
    printf("Количество четных чисел: %d", ct_even);
    if (ct_even > 0)
        printf("  среднее значение: %g", sumeven / ct_even);
    putchar('\n');
    printf("Количество нечетных чисел: %d", ct_odd);
    if (ct_odd > 0)
        printf("  среднее значение: %g", sumodd / ct_odd);
    putchar('\n');
    printf("\nПрограмма завершена.\n");
    return 0;
}
```

7.5.

```
/* Упражнение по программированию 7.5 */
#include <stdio.h>
int main(void)
{
    char ch;
    int ct1 = 0;
    int ct2 = 0;
    while ((ch = getchar()) != '#')
    {
```

```
        switch(ch)
        {
            case '.' : putchar('!');
                       ++ct1;
                       break;
            case '!' : putchar('!');
                       putchar('!');
                       ++ct2;
                       break;
            default : putchar(ch);
        }
    }
    printf("%d замен . символом !\n", ct1);
    printf("%d замен! символом !!\n", ct2);
    return 0;
}
```

7.7.

```
// Упражнение по программированию 7.7
#include <stdio.h>
#define BASEPAY 10    // $10 в час
#define BASEHRS 40    // часов по основной тарифной ставке
#define OVERTIME 1.5  // в 1.5 раза больше
#define AMT1 300      // 1-й уровень тарифов
#define AMT2 150      // 2-й уровень тарифов
#define RATE1 0.15    // ставка для 1-го уровня
#define RATE2 0.20    // ставка для 2-го уровня
#define RATE3 0.25    // ставка для 3-го уровня
int main(void)
{
    double hours;
    double gross;
    double net;
    double taxes;
    printf("Введите количество часов, отработанных за эту неделю: ");
    scanf("%lf", &hours);
    if (hours <= BASEHRS)
        gross = hours * BASEPAY;
    else
        gross = BASEHRS * BASEPAY + (hours - BASEHRS) * BASEPAY * OVERTIME;
    if (gross <= AMT1)
        taxes = gross * RATE1;
    else if (gross <= AMT1 + AMT2)
        taxes = AMT1 * RATE1 + (gross - AMT1) * RATE2;
    else
        taxes = AMT1 * RATE1 + AMT2 * RATE2 + (gross - AMT1 - AMT2) * RATE3;
    net = gross - taxes;
    printf("общая сумма: $%.2f; налоги: $%.2f; чистый доход: $%.2f\n", gross,
taxes, net);
    return 0;
}
```

7.9.

```
/* Упражнение по программированию 7.9 */
#include <stdio.h>
#include <stdbool.h>
int main(void)
{
```

```
    int limit;
    int num;
    int div;
    bool numIsPrime; // использовать int в случае недоступности stdbool.h
    printf("Введите положительное целое число: ");
    while (scanf("%d", &limit) == 1 && limit > 0)
    {
       if (limit > 1)
          printf("Простые числа вплоть до %d\n", limit);
       else
          printf("Простых чисел нет.\n");
       for (num = 2; num <= limit; num++)
       {
          for (div = 2, numIsPrime = true; (div * div) <= num; div++)
             if (num % div == 0)
                numIsPrime = false;
          if (numIsPrime)
             printf("%d - простое число.\n", num);
       }
       printf("Введите положительное целое число (или q для завершения): ");
    }
    printf("Программа завершена.\n");
    return 0;
}
```

7.11.

```
/* pe7-11.c */
/* Упражнение по программированию 7.11 */
#include <stdio.h>
#include <ctype.h>
int main(void)
{
    const double price_artichokes = 2.05;
    const double price_beets = 1.15;
    const double price_carrots = 1.09;
    const double DISCOUNT_RATE = 0.05;
    const double under5 = 6.50;
    const double under20 = 14.00;
    const double base20 = 14.00;
    const double extralb = 0.50;
    char ch;
    double lb_artichokes = 0;
    double lb_beets = 0;
    double lb_carrots = 0;
    double lb_temp;
    double lb_total;
    double cost_artichokes;
    double cost_beets;
    double cost_carrots;
    double cost_total;
    double final_total;
    double discount;
    double shipping;
    printf("Введите a для покупки артишоков, b для покупки свеклы, ");
    printf("c для покупки моркови, q для выхода: ");
    while ((ch = getchar()) != 'q' && ch != 'Q')
    {
       if (ch == '\n')
          continue;
```

```
        while (getchar() != '\n')
            continue;
        ch = tolower(ch);
        switch (ch)
        {
            case 'a' : printf("Введите желаемое количество фунтов артишоков: ");
                       scanf("%lf", &lb_temp);
                       lb_artichokes += lb_temp;
                       break;
            case 'b' : printf("Введите желаемое количество фунтов свеклы: ");
                       scanf("%lf", &lb_temp);
                       lb_beets += lb_temp;
                       break;
            case 'c' : printf("Введите желаемое количество фунтов моркови: ");
                       scanf("%lf", &lb_temp);
                       lb_carrots += lb_temp;
                       break;
            default  : printf("%c не является допустимым вариантом.\n", ch);
        }
        printf("Введите a для покупки артишоков, b для покупки свеклы, ");
        printf("c для покупки моркови, q для выхода: ");
    }
    cost_artichokes = price_artichokes * lb_artichokes;
    cost_beets = price_beets * lb_beets;
    cost_carrots = price_carrots * lb_carrots;
    cost_total = cost_artichokes + cost_beets + cost_carrots;
    lb_total = lb_artichokes + lb_beets + lb_carrots;
    if (lb_total <= 0)
        shipping = 0.0;
    else if (lb_total < 5.0)
        shipping = under5;
    else if (lb_total < 20)
        shipping = under20;
    else
        shipping = base20 + extralb * lb_total;
    if (cost_total > 100.0)
        discount = DISCOUNT_RATE * cost_total;
    else
        discount = 0.0;
    final_total = cost_total + shipping - discount;
    printf("Ваш заказ:\n");
    printf("%.2f фунтов артишоков по $%.2f за фунт: $%.2f\n",
           lb_artichokes, price_artichokes, cost_artichokes);
    printf("%.2f фунтов свеклы по $%.2f за фунт: $%.2f\n",
           lb_beets, price_beets, cost_beets);
    printf("%.2f фунтов моркови по $%.2f за фунт: $%.2f\n",
           lb_carrots, price_carrots, cost_carrots);
    printf("Суммарная стоимость овощей: $%.2f\n", cost_total);
    if (cost_total > 100)
        printf("Скидка: $%.2f\n", discount);
    printf("Доставка: $%.2f\n", shipping);
    printf("Общие расходы: $%.2f\n", final_total);
    return 0;
}
```

Упражнения по программированию из главы 8

8.1.

```
/* Упражнение по программированию 8.1 */
#include <stdio.h>
int main(void)
{
   int ch;
   int ct = 0;
   while ((ch = getchar()) != EOF)
      ct++;
   printf("Количество прочитанных символов: %d\n", ct);
   return 0;
}
```

8.3.

```
/* Упражнение по программированию 8.3 */
/* Использование ctype.h устраняет необходимость в предположении
последовательного кодирования. */
#include <stdio.h>
#include <ctype.h>
int main(void)
{
   int ch;
   unsigned long uct = 0;
   unsigned long lct = 0;
   unsigned long oct = 0;
   while ((ch = getchar()) != EOF)
      if (isupper(ch))
         uct++;
      else if (islower(ch))
         lct++;
      else
         oct++;
   printf("Количество прочитанных прописных символов: %lu\n", uct);
   printf("Количество прочитанных строчных символов: %lu\n", lct);
   printf("Количество прочитанных остальных символов: %lu\n", oct);
   return 0;
}
/*
 или же можно использовать такой код:
 if (ch >= 'A' && ch <= 'Z')
    uct++;
 else if (ch >= 'a' && ch <= 'z')
    lct++;
 else
    oct++;
*/
```

8.5.

```
/* Упражнение по программированию 8.5 */
/* binaryguess.c -- усовершенствованная программа угадывания чисел, */
/* полагающаяся на правдивые и корректные ответы                    */
#include <stdio.h>
#include <ctype.h>
int main(void)
{
```

```
    int high = 100;
    int low = 1;
    int guess = (high + low) / 2;
    char response;
    printf("Выберите целое число в интервале от 1 до 100. Я попробую угадать ");
    printf("его.\nНажмите клавишу y, если моя догадка верна,");
    printf("\nклавишу h, если число больше, и клавишу l, если оно меньше.\n");
    printf("Вашим числом является %d?\n", guess);
    while ((response = getchar()) != 'y') /* получить ответ */
    {
       if (response == '\n')
          continue;
       if (response != 'h' && response != 'l')
       {
          printf("Я не понимаю этот ответ. Нажмите h, если число больше,\n");
          printf("l - если меньше или y, если число угадано.\n");
          continue;
       }
       if (response == 'h')
          high = guess - 1;
       else if (response == 'l')
          low = guess + 1;
       guess = (high + low) / 2;
       printf("Ладно, тогда это %d?\n", guess);
    }
    printf("Я знал, что у меня получится!\n");
    return 0;
}
```

8.7.

```
/* Упражнение по программированию 8.7 */
#include <stdio.h>
#include <ctype.h>
#include <stdio.h>
#define BASEPAY1 8.75     // $8.75 в час
#define BASEPAY2 9.33     // $9.33 в час
#define BASEPAY3 10.00    // $10.00 в час
#define BASEPAY4 11.20    // $11.20 в час
#define BASEHRS 40        // часов по основной тарифной ставке
#define OVERTIME 1.5      // в 1.5 раза больше
#define AMT1 300          // 1-й уровень тарифов
#define AMT2 150          // 2-й уровень тарифов
#define RATE1 0.15        // ставка для 1-го уровня
#define RATE2 0.20        // ставка для 2-го уровня
#define RATE3 0.25        // ставка для 3-го уровня
int getfirst(void);
void menu(void);
int main(void)
{
   double hours;
   double gross;
   double net;
   double taxes;
   double pay;
   char response;
   menu();
   while ((response = getfirst()) != 'q')
   {
```

```
      if (response == '\n') /* пропустить символы новой строки */
         continue;
      response = tolower(response); /* принимать A как a и т.д. */
      switch (response)
      {
         case 'a': pay = BASEPAY1; break;
         case 'b': pay = BASEPAY2; break;
         case 'c': pay = BASEPAY3; break;
         case 'd': pay = BASEPAY4; break;
         default : printf("Введите a, b, c, d или q.\n");
                   menu();
                   continue; // перейти в начало цикла
      }
      printf("Введите количество часов, отработанных за эту неделю: ");
      scanf("%lf", &hours);
      if (hours <= BASEHRS)
         gross = hours * pay;
      else
         gross = BASEHRS * pay + (hours - BASEHRS) * pay * OVERTIME;
      if (gross <= AMT1)
         taxes = gross * RATE1;
      else if (gross <= AMT1 + AMT2)
         taxes = AMT1 * RATE1 + (gross - AMT1) * RATE2;
      else
         taxes = AMT1 * RATE1 + AMT2 * RATE2 + (gross - AMT1 - AMT2) * RATE3;
      net = gross - taxes;
      printf("общая сумма: $%.2f; налоги: $%.2f; чистый доход: $%.2f\n", gross,
             taxes, net);
      menu();
   }
   printf("Программа завершена.\n");
   return 0;
}

void menu(void)
{
   printf("*******************************************************"
         "*********\n");
   printf("Введите букву, соответствующую желаемой тарифной ставке"
          " или действию:\n");
   printf("a) $%4.2f/ч b) $%4.2f/ч\n", BASEPAY1,
          BASEPAY2);
   printf("c) $%5.2f/ч d) $%5.2f/ч\n", BASEPAY3,
          BASEPAY4);
   printf("q) Выход\n");
   printf("*******************************************************"
          "*********\n");
}

int getfirst(void)
{
   int ch;
   ch = getchar();
   while (isspace(ch))
      ch = getchar();
   while (getchar() != '\n')
      continue;
   return ch;
}
```

Упражнения по программированию из главы 9

9.1.

```
/* Упражнение по программированию 9.1 */
#include <stdio.h>
double min(double, double);
int main(void)
{
   double x, y;
   printf("Введите два числа (или q для завершения): ");
   while (scanf("%lf %lf", &x, &y) == 2)
   {
      printf("Меньшим числом является %f.\n", min(x,y));
      printf("Введите следующие два числа (или q для завершения): ");
   }
   printf("Программа завершена.\n");
   return 0;
}

double min(double a, double b)
{
   return a < b ? a : b;
}

/* альтернативная реализация
 double min(double a, double b)
 {
    if (a < b)
       return a;
    else
       return b;
    }
*/
```

9.3.

```
/* Упражнение по программированию 9.3 */
#include <stdio.h>
void chLineRow(char ch, int c, int r);
int main(void)
{
   char ch;
   int col, row;
   printf("Введите символ (# для завершения): ");
   while ( (ch = getchar()) != '#')
   {
      if (ch == '\n')
         continue;
      printf("Введите количество столбцов и количество строк: ");
      if (scanf("%d %d", &col, &row) != 2)
         break;
      chLineRow(ch, col, row);
      printf("\nВведите следующий символ (# для завершения): ");
   }
   printf("Программа завершена.\n");
   return 0;
}

// строки и столбцы начинаются с 0
void chLineRow(char ch, int c, int r)
```

```
{
   int col, row;
   for (row = 0; row < r ; row++)
   {
      for (col = 0; col < c; col++)
         putchar(ch);
      putchar('\n');
   }
   return;
}
```

9.5.

```
/* Упражнение по программированию 9.5 */
#include <stdio.h>
void larger_of(double *p1, double *p2);
int main(void)
{
   double x, y;
   printf("Введите два числа (или q для завершения): ");
   while (scanf("%lf %lf", &x, &y) == 2)
   {
      larger_of(&x, &y);
      printf("Модифицированными значениями являются %f и %f.\n", x, y);
      printf("Введите следующие два числа (или q для завершения): ");
   }
   printf("Программа завершена.\n");
   return 0;
}

void larger_of(double *p1, double *p2)
{
   if (*p1 > *p2)
      *p2 = *p1;
   else
      *p1 = *p2;
}

// альтернативная реализация:
/*
 void larger_of(double *p1, double *p2)
 {
    *p1= *p2 = *p1 > *p2 ? *p1 : *p2;
 }
*/
```

9.8.

```
/* Упражнение по программированию 9.8 */
#include <stdio.h>
double power(double a, int b); /* прототип ANSI */
int main(void)
{
   double x, xpow;
   int n;
   printf("Введите число и целочисленную степень,");
   printf(" в которую\nчисло будет возведено. Либо введите q");
   printf(" для завершения.\n");
   while (scanf("%lf%d", &x, &n) == 2)
   {
```

```
        xpow = power(x,n); /* вызов функции */
        printf("%.3g в степени %d равно %.5g\n", x, n, xpow);
        printf("Введите следующую пару чисел или q для завершения.\n");
    }
    printf("Программа завершена.\n");
    return 0;
}
double power(double a, int b) /* определение функции */
{
    double pow = 1;
    int i;
    if (b == 0)
    {
        if (a == 0)
            printf("Результат возведения 0 в степень 0 не определен; используйте 1
в качестве значения\n");
        pow = 1.0;
    }
    else if (a == 0)
        pow = 0.0;
    else if (b > 0)
        for(i = 1; i <= b; i++)
            pow *= a;
    else /* b < 0 */
        pow = 1.0 / power(a, - b);
    return pow; /* возвратить значение pow */
}
```

9.10.

```
/* Упражнение по программированию 9.10 */
#include <stdio.h>
void to_base_n(int x, int base);
int main(void)
{
    int number;
    int b;
    int count;
    printf("Введите целое число (или q для завершения):\n");
    while (scanf("%d", &number) == 1)
    {
        printf("Введите основание системы счисления (2-10): ");
        while ((count = scanf("%d", &b))== 1
               && (b < 2 || b > 10))
        {
            printf("Основание системы счисления должно находиться в диапазоне 2-10: ");
        }
        if (count != 1)
            break;
        printf("Эквивалентная запись по основанию %d: ", b);
        to_base_n(number, b);
        putchar('\n');
        printf("Введите целое число (или q для завершения):\n");
    }
    printf("Программа завершена.\n");
    return 0;
}
```

```
void to_base_n(int x, int base) /* рекурсивная функция */
{
   int r;
   r = x % base;
   if (x >= base)
      to_base_n(x / base, base);
   putchar('0' + r);
   return;
}
```

Упражнения по программированию из главы 10

10.1.

```
/* Упражнение по программированию 10.1 */
#include <stdio.h>
#define MONTHS 12     // количество месяцев в году
#define YRS 5         // количество лет, для которых доступны данные
int main(void)
{
   // инициализация данными об осадках за период с 2010 по 2014 гг.
   const float rain[YRS][MONTHS] = {
      {4.3,4.3,4.3,3.0,2.0,1.2,0.2,0.2,0.4,2.4,3.5,6.6},
      {8.5,8.2,1.2,1.6,2.4,0.0,5.2,0.9,0.3,0.9,1.4,7.3},
      {9.1,8.5,6.7,4.3,2.1,0.8,0.2,0.2,1.1,2.3,6.1,8.4},
      {7.2,9.9,8.4,3.3,1.2,0.8,0.4,0.0,0.6,1.7,4.3,6.2},
      {7.6,5.6,3.8,2.8,3.8,0.2,0.0,0.0,0.0,1.3,2.6,5.2}
   };
   int year, month;
   float subtot, total;
   printf("ГОД   КОЛИЧЕСТВО ОСАДКОВ (в дюймах)\n");
   for (year = 0, total = 0; year < YRS; year++)
   { /* для каждого года суммировать количество осадков за каждый месяц */
      for (month = 0, subtot = 0; month < MONTHS; month++)
         subtot += *(*(rain + year) + month);
      printf("%5d %15.1f\n", 2010 + year, subtot);
      total += subtot; /* общая сумма для всех лет */
   }
   printf("\nСреднегодовое количество осадков составляет %.1f дюймов.\n\n", total/YRS);
   printf("СРЕДНЕМЕСЯЧНОЕ КОЛИЧЕСТВО ОСАДКОВ:\n\n");
   printf("Янв  Фев  Мар  Апр  Май  Июн  Июл  Авг  Сен  Окт");
   printf("  Ноя  Дек\n");
   for (month = 0; month < MONTHS; month++)
   { /* для каждого месяца суммировать количество осадков на протяжении годов */
      for (year = 0, subtot =0; year < YRS; year++)
         subtot += *(*(rain + year) + month);
      printf("%4.1f ", subtot/YRS);
   }
   printf("\n");
   return 0;
}
```

10.3.

```
/* Упражнение по программированию 10.3 */
#include <stdio.h>
#define LEN 10
int max_arr(const int ar[], int n);
void show_arr(const int ar[], int n);
```

```
int main(void)
{
   int orig[LEN] = {1,2,3,4,12,6,7,8,9,10};
   int max;
   show_arr(orig, LEN);
   max = max_arr(orig, LEN);
   printf("%d = наибольшее значение\n", max);
   return 0;
}

int max_arr(const int ar[], int n)
{
   int i;
   int max = ar[0];
   /* не используйте 0 в качестве начального значения max -- не сработает,
      если все значения в массиве являются отрицательными */
   for (i = 1; i < n; i++)
      if (max < ar[i])
         max = ar[i];
   return max;
}

void show_arr(const int ar[], int n)
{
   int i;
   for (i = 0; i < n; i++)
      printf("%d ", ar[i]);
   putchar('\n');
}
```

10.5.

```
/* Упражнение по программированию 10.5 */
#include <stdio.h>
#define LEN 10
double max_diff(const double ar[], int n);
void show_arr(const double ar[], int n);
int main(void)
{
   double orig[LEN] = {1.1,2,3,4,12,61.3,7,8,9,10};
   double max;
   show_arr(orig, LEN);
   max = max_diff(orig, LEN);
   printf("%g = максимальная разность\n", max);
   return 0;
}

double max_diff(const double ar[], int n)
{
   int i;
   double max = ar[0];
   double min = ar[0];
   for (i = 1; i < n; i++)
   {
      if (max < ar[i])
         max = ar[i];
      else if (min > ar[i])
         min = ar[i];
   }
   return max - min;
}
```

```
void show_arr(const double ar[], int n)
{
   int i;
   for (i = 0; i < n; i++)
      printf("%g ", ar[i]);
   putchar('\n');
}
```

10.8.

```
/* Упражнение по программированию 10.8 */
#include <stdio.h>
#define LEN1 7
#define LEN2 3
void copy_arr(int ar1[], const int ar2[], int n);
void show_arr(const int [], int);
int main(void)
{
   int orig[LEN1] = {1,2,3,4,5,6,7};
   int copy[LEN2];
   show_arr(orig, LEN1);
   copy_arr(copy, orig + 2, LEN2);
   show_arr(copy, LEN2);
   return 0;
}
void copy_arr(int ar1[], const int ar2[], int n)
{
   int i;
   for (i = 0; i < n; i++)
      ar1[i] = ar2[i];
}
void show_arr(const int ar[], int n)
{
   int i;
   for (i = 0; i < n; i++)
      printf("%d ", ar[i]);
   putchar('\n');
}
```

10.11.

```
/* Упражнение по программированию 10.11 */
#include <stdio.h>
#define ROWS 3
#define COLS 5
void times2(int ar[][COLS], int r);
void showarr2(int ar[][COLS], int r);
int main(void)
{
   int stuff[ROWS][COLS] = { {1,2,3,4,5},
                             {6,7,8,-2,10},
                             {11,12,13,14,15}
                           };
   showarr2(stuff, ROWS);
   putchar('\n');
   times2(stuff, ROWS);
   showarr2(stuff, ROWS);
   return 0;
}
```

```
void times2(int ar[][COLS], int r)
{
   int row, col;
   for (row = 0; row < r; row++)
      for (col = 0; col < COLS; col++)
         ar[row][col] *= 2;
}

void showarr2(int ar[][COLS], int r)
{
   int row, col;
   for (row = 0; row < r; row++)
   {
      for (col = 0; col < COLS; col++)
         printf("%d ", ar[row][col]);
      putchar('\n');
   }
}
```

10.14.

```
/* Упражнение по программированию 10.14 */
#include <stdio.h>
#define ROWS 3
#define COLS 5
void store(double ar[], int n);
double average2d(int rows, int cols, double ar[rows][cols]);
double max2d(int rows, int cols, double ar[rows][cols]);
void showarr2(int rows, int cols, double ar[rows][cols]);
double average(const double ar[], int n);
int main(void)
{
   double stuff[ROWS][COLS];
   int row;
   for (row = 0; row < ROWS; row++)
   {
      printf("Введите %d чисел для строки %d\n", COLS, row + 1);
      store(stuff[row], COLS);
   }
   printf("содержимое массива:\n");
   showarr2(ROWS, COLS, stuff);
   for (row = 0; row < ROWS; row++)
      printf("среднее значение для строки %d = %g\n", row + 1,
             average(stuff[row], COLS));
   printf("среднее значение для всех строк = %g\n", average2d(ROWS, COLS, stuff));
   printf("наибольшее значение = %g\n", max2d(ROWS, COLS, stuff));
   printf("Программа завершена.\n");
   return 0;
}

void store(double ar[], int n)
{
   int i;
   for (i = 0; i < n; i++)
   {
      printf("Введите значение #%d: ", i + 1);
      scanf("%lf", & ar[i]);
   }
}
```

```
double average2d(int rows, int cols, double ar[rows][cols])
{
    int r, c;
    double sum = 0.0;
    for (r = 0; r < rows; r++)
        for (c = 0; c < cols; c++)
            sum += ar[r][c];
    if (rows * cols > 0)
        return sum / (rows * cols);
    else
        return 0.0;
}

double max2d(int rows, int cols, double ar[rows][cols])
{
    int r, c;
    double max = ar[0][0];
    for (r = 0; r < rows; r++)
        for (c = 0; c < cols; c++)
            if (max < ar[r][c])
    max = ar[r][c];
    return max;
}

void showarr2(int rows, int cols, double ar[rows][cols])
{
    int row, col;
    for (row = 0; row < rows; row++)
    {
        for (col = 0; col < cols; col++)
            printf("%g ", ar[row][col]);
        putchar('\n');
    }
}

double average(const double ar[], int n)
{
    int i;
    double sum = 0.0;
    for (i = 0; i < n; i++)
        sum += ar[i];
    if (n > 0)
        return sum / n;
    else
        return 0.0;
}
```

Упражнения по программированию из главы 11

11.1.

```
/* Упражнение по программированию 11.1 */
#include <stdio.h>
#define LEN 10
char * getnchar(char * str, int n);
int main(void)
{
    char input[LEN];
    char *check;
    check = getnchar(input, LEN - 1);
```

```
    if (check == NULL)
        puts("Сбой при вводе.");
    else
        puts(input);
    puts("Программа завершена.\n");
    return 0;
}
char * getnchar(char * str, int n)
{
    int i;
    int ch;
    for (i = 0; i < n; i++)
    {
        ch = getchar();
        if (ch != EOF)
            str[i] = ch;
        else
            break;
    }
    if (ch == EOF)
        return NULL;
    else
    {
        str[i] = '\0';
        return str;
    }
}
```

11.3.

```
/* Упражнение по программированию 11.3 */
#include <stdio.h>
#define LEN 80
char * getword(char * str);
int main(void)
{
    char input[LEN];
    while (getword(input) != NULL)
        puts(input);
    puts("Программа завершена.\n");
    return 0;
}
#include <ctype.h>
char * getword(char * str)
{
    int ch;
    char * orig = str;
    // пропустить начальные пробельные символы
    while ((ch = getchar()) != EOF && isspace(ch))
        continue;
    if (ch == EOF)
        return NULL;
    else
        *str++ = ch; // первый символ в слове
    // получить остаток слова
    while ((ch = getchar()) != EOF && !isspace(ch))
        *str++ = ch;
    *str = '\0';
```

```
    if (ch == EOF)
        return NULL;
    else
    {
        while (ch != '\n')
            ch = getchar();
        return orig;
    }
}
```

11.6.

```
/* Упражнение по программированию 11.6 */
#include <stdio.h>
#include <string.h>
#define LEN 80
_Bool is_within(const char * str, char c);
char * s_gets(char * st, int n);
int main(void)
{
    char input[LEN];
    char ch;
    int found;;
    printf("Введите строку: ");
    while (s_gets(input, LEN) && input[0] != '\0')
    {
        printf("Введите символ: ");
        ch = getchar();
        while (getchar() != '\n')
            continue;
        found = is_within(input, ch);
        if (found == 0)
            printf("%c не найден в строке.\n", ch);
        else
            printf("%c найден в строке %s\n", ch, input);
        printf("Следующая строка: ");
    }
    puts("Программа завершена.\n");
    return 0;
}

_Bool is_within(const char * str, char ch)
{
    while (*str != ch && *str != '\0')
        str++;
    return *str; /* равно 0, если достигнут символ \0, и ненулевому значению в
противном случае */
}

char * s_gets(char * st, int n)
{
    char * ret_val;
    char * find;
    ret_val = fgets(st, n, stdin);
    if (ret_val)
    {
        find = strchr(st, '\n');  // поиск новой строки
        if (find)                 // если адрес не равен NULL,
            *find = '\0';         // поместить туда нулевой символ
```

```
        else
            while (getchar() != '\n')
                continue;       // отбросить остаток строки
    }
    return ret_val;
}
```

11.8.

```
/* Упражнение по программированию 11.8 */
#include <stdio.h>
#define LEN 20
char * string_in(const char * s1, const char * s2);
int main(void)
{
    char orig[LEN] = "transportation";
    char * find;
    puts(orig);
    find = string_in(orig, "port");
    if (find)
        puts(find);
    else
        puts("Не найдено");
    find = string_in(orig, "part");
    if (find)
        puts(find);
    else
        puts("Не найдено");
    return 0;
}
#include <string.h>
char * string_in(const char * s1, const char * s2)
{
    int l2 = strlen(s2);
    int tries;       /* максимальное количество сравнений        */
    int nomatch = 1; /* установить в 0, если соответствие найдено */
    tries = strlen(s1) + 1 - l2;
    if (tries > 0)
        while (( nomatch = strncmp(s1, s2, l2)) && tries--)
            s1++;
    if (nomatch)
        return NULL;
    else
        return (char *) s1; /* приведение */
}
```

11.10.

```
/* Упражнение по программированию 11.10 */
#include <stdio.h>
#include <string.h> // для strchr();
#define LEN 81
int drop_space(char * s);
char * s_gets(char * st, int n);
int main(void)
{
    char orig[LEN];
    puts("Введите строку, содержащую 80 символов или меньше:");
```

```
    while (s_gets(orig, LEN) && orig[0] != '\0')
    {
        drop_space(orig);
        puts(orig);
        puts("Введите следующую строку (или просто нажмите Enter для завершения):");
    }
    puts("Программа завершена.");
    return 0;
}
int drop_space(char * s)
{
    char * pos;
    while (*s) /* or while (*s != '\0') */
    {
        if (*s == ' ')
        {
            pos = s;
            do
            {
                *pos = *(pos + 1);
                pos++;
            } while (*pos);
        }
        else
            s++;
    }
}
char * s_gets(char * st, int n)
{
    char * ret_val;
    char * find;
    ret_val = fgets(st, n, stdin);
    if (ret_val)
    {
        find = strchr(st, '\n');  // поиск новой строки
        if (find)                 // если адрес не равен NULL,
            *find = '\0';         // поместить туда нулевой символ
        else
            while (getchar() != '\n')
                continue;         // отбросить остаток строки
    }
    return ret_val;
}
```

11.12.

```
/* pe11-12.c -- подсчет слов и определенных символов */
/* Упражнение по программированию 11.11 */
#include <stdio.h>
#include <ctype.h>    // для isspace()
#include <stdbool.h>  // для bool, true, false
int main(void)
{
    char c;           // прочитанный символ
    int low_ct = 0;   // количество строчных символов
    int up_ct = 0;    // количество прописных символов
    int dig_ct = 0;   // количество цифр
    int n_words = 0;  // количество слов
```

```
    int punc_ct = 0;       // количество знаков препинания
    bool inword = false;  // == true, если c находится в слове
    printf("Введите текст, подлежащий анализу (EOF для завершения):\n");
    while ((c = getchar()) != EOF)
    {
       if (islower(c))
          low_ct++;
       else if (isupper(c))
          up_ct++;
       else if (isdigit(c))
          dig_ct++;
       else if (ispunct(c))
          punc_ct++;
       if (!isspace(c) && !inword)
       {
          inword = true;  // начало нового слова
          n_words++;      // подсчет слов
       }
       if (isspace(c) && inword)
          inword = false; // достигнут конец слова
    }
    printf("\nслов = %d, строчных = %d, прописных = %d, "
          "цифр = %d, препинаний = %d\n",
          n_words,low_ct,up_ct, dig_ct, punc_ct);
    return 0;
}
```

11.14.

```
/* Упражнение по программированию 11.14 */
#include <stdio.h>
#include <stdlib.h> /* для atof() */
#include <math.h>   /* для pow()   */
int main(int argc, char *argv[])
{
   double num, exp;
   if (argc != 3)
      printf("Использование: %s число экспонента\n", argv[0]);
   else
   {
      num = atof(argv[1]);
      exp = atof(argv[2]);
      printf("%f в степени %f = %g\n", num, exp, pow(num,exp));
   }
   return 0;
}
```

11.16.

```
/* Упражнение по программированию 11.16 */
#include <stdio.h>
#include <ctype.h>
int main(int argc, char *argv[])
{
   char mode = 'p';
   int ok = 1;
   int ch;
   if (argc > 2)
   {
      printf("Использование: %s [-p | -u | -l]\n", argv[0]);
```

```
        ok = 0; /* пропустить обработку входных данных */
    }
    else if (argc == 2)
    {
        if (argv[1][0] != '-')
        {
            printf("Использование: %s [-p | -u | -l]\n", argv[0]);
            ok = 0;
        }
        else
            switch(argv[1][1])
            {
                case 'p' :
                case 'u' :
                case 'l' : mode = argv[1][1];
                           break;
                default  : printf("%s является некорректным флагом; ", argv[1]);
                           printf("используется стандартный флаг (-p).\n");
            }
    }
    if (ok)
        while ((ch = getchar() ) != EOF)
        {
            switch(mode)
            {
                case 'p' : putchar(ch);
                           break;
                case 'u' : putchar(toupper(ch));
                           break;
                case 'l' : putchar(tolower(ch));
            }
        }
    return 0;
}
```

Упражнения по программированию из главы 12

12.1.

```
/* pe12-1.c -- устранение глобальных переменных из global.c */
/* Упражнение по программированию 12.1 */
/* Один из нескольких подходов */
#include <stdio.h>
void critic(int * u);
int main(void)
{
    int units; /* переменная units теперь локальная */
    printf("Сколько фунтов весит маленький бочонок масла?\n");
    scanf("%d", &units);
    while ( units != 56)
        critic(&units);
    printf("Вы знали это!\n");
    return 0;
}
void critic(int * u)
{
    printf("Вам не повезло. Попробуйте еще раз.\n");
    scanf("%d", u);
}
```

```
// либо используйте возвращаемое значение:
// units = critic();
// и приведите функцию critic() к следующему виду:
/*
 int critic(void)
 {
    int u;
    printf("Вам не повезло. Попробуйте еще раз.\n");
    scanf("%d", &u);
    return u;
 }
*/
// либо обеспечьте в main() ввод следующего значения для units
```

12.3.

```
// pe12-3a.h
#define METRIC 0
#define US 1
#define USE_RECENT 2
void check_mode(int *pm);
void get_info(int mode, double * pd, double * pf);
void show_info(int mode, double distance, double fuel);
```

```
// pe12-3a.c
// компилировать вместе с pe12-3b.c
#include <stdio.h>
#include "pe12-3a.h"
void check_mode(int *pm)
{
   if (*pm != METRIC && *pm != US)
   {
      printf("Указан недопустимый режим. Режим %d\n", *pm);
      printf("Будет использоваться предыдущий режим.\n");
      *pm = USE_RECENT;
   }
}

void get_info(int mode, double * pd, double * pf)
{
   if (mode == METRIC)
      printf("Введите пройденное расстояние в километрах: ");
   else
      printf("Введите пройденное расстояние в милях: ");
   scanf("%lf",pd);
   if (mode == METRIC)
      printf("Введите объем израсходованного топлива в литрах: ");
   else
      printf("Введите объем израсходованного топлива в галлонах: ");
   scanf("%lf", pf);
}

void show_info(int mode, double distance, double fuel)
{
   printf("Расход топлива составляет ");
   if (mode == METRIC)
      printf("%.2f литров на 100 км.\n", 100 * fuel / distance);
   else
      printf("%.1f мили на галлон.\n", distance / fuel);
}
```

```
// pe12-3b.c
// compile with pe12-3a.c
#include <stdio.h>
#include "pe12-3a.h"
int main(void)
{
   int mode;
   int prev_mode = METRIC;
   double distance, fuel;
   printf("Введите 0 для метрического режима или 1 для американского режима: ");
   scanf("%d", &mode);
   while (mode >= 0)
   {
      check_mode(&mode);
      if (mode == USE_RECENT)
         mode = prev_mode;
      prev_mode = mode;
      get_info(mode, &distance, &fuel);
      show_info(mode, distance, fuel);
      printf("Введите 0 для метрического режима или 1 для американского режима");
      printf(" (-1 для завершения): ");
      scanf("%d", &mode);
   }
   printf("Программа завершена.\n");
   return 0;
}
```

12.5.

```
/* pe12-5.c */
#include <stdio.h>
#include <stdlib.h>
void print(const int array[], int limit);
void sort(int array[], int limit);
#define SIZE 100
int main(void)
{
   int i;
   int arr[SIZE];
   for (i = 0; i < SIZE; i++)
      arr[i] = rand() % 10 + 1;
   puts("начальный массив");
   print(arr,SIZE);
   sort(arr,SIZE);
   puts("\nотсортированный массив");
   print(arr,SIZE);
   return 0;
}
/* sort.c -- сортировка целочисленного массива в убывающем порядке */
void sort(int array[], int limit)
{
   int top, search, temp;
   for (top = 0; top < limit -1; top++)
      for (search = top + 1; search < limit; search++)
         if (array[search] > array[top])
         {
            temp = array[search];
            array[search] = array[top];
            array[top] = temp;
         }
}
```

```
/* print.c -- вывод массива */
void print(const int array[], int limit)
{
   int index;
   for (index = 0; index < limit; index++)
   {
      printf("%2d ", array[index]);
      if (index % 10 == 9)
         putchar('\n');
   }
   if (index % 10 != 0) // если последняя строка вывода не заполнена
      putchar('\n');
}
```

12.7.

```
/* pe12-7.c */
#include <stdio.h>
#include <stdlib.h> /* для srand() */
#include <time.h>   /* для time()  */
int rollem(int);
int main(void)
{
   int dice, count, roll;
   int sides;
   int set, sets;
   srand((unsigned int) time(0)); /* рандомизировать rand() */
   printf("Введите количество бросаний или q для завершения: ");
   while (scanf("%d", &sets) == 1)
   {
      printf("Сколько граней и сколько костей? ");
      if (scanf("%d %d", &sides, &dice) != 2)
      {
         puts("не целые числа -- завершение цикла ввода.");
         break;
      }
      printf("Имеем %d бросаний %d костей с %d гранями.\n", sets, dice, sides);
      for (set = 0; set < sets; set++)
      {
         for (roll = 0, count = 0; count < dice; count++)
            roll += rollem(sides);
         /* промежуточная сумма точек костей */
         printf("%4d ", roll);
         if (set % 15 == 14)
            putchar('\n');
      }
      if (set % 15 != 0)
         putchar('\n');
      printf("Введите количество бросаний или q для завершения: ");
   }
   puts("Пусть удача не покидает вас!\n");
   return 0;
}

int rollem(int sides)
{
   int roll;
   roll = rand() % sides + 1;
   return roll;
}
```

Упражнения по программированию из главы 13

13.2.

```
/* Упражнение по программированию 13.2 */
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
   int byte;
   FILE * source;
   FILE * target;
   if (argc != 3)
   {
      printf("Использование: %s исходный-файл целевой-файл\n", argv[0]);
      exit(EXIT_FAILURE);
   }
   if ((source = fopen(argv[1], "rb")) == NULL)
   {
      printf("Не удается открыть файл %s для ввода\n", argv[1]);
      exit(EXIT_FAILURE);
   }
   if ((target = fopen(argv[2], "wb")) == NULL)
   {
      printf("Не удается открыть файл %s для вывода\n", argv[2]);
      exit(EXIT_FAILURE);
   }
   while ((byte = getc(source)) != EOF)
   {
      putc(byte, target);
   }
   if (fclose(source) != 0)
      printf("Не удается закрыть файл %s\n", argv[1]);
   if (fclose(target) != 0)
      printf("Не удается закрыть файл %s\n", argv[2]);
   return 0;
}
```

13.4.

```
/* Упражнение по программированию 13.4 */
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
   int byte;
   FILE * source;
   int filect;
   if (argc == 1)
   {
      printf("Использование: %s имя-файла[имена-файлов]\n", argv[0]);
      exit(EXIT_FAILURE);
   }
   for (filect = 1; filect < argc; filect++)
   {
      if ((source = fopen(argv[filect], "r")) == NULL)
      {
         printf("Не удается открыть файл %s для ввода\n", argv[filect]);
         continue;
      }
```

```
        while ((byte = getc(source)) != EOF)
        {
            putchar(byte);
        }
        if (fclose(source) != 0)
            printf("Не удается закрыть файл %s\n", argv[1]);
    }
    return 0;
}
```

13.5.

```
/* Упражнение по программированию 13.5 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define BUFSIZE 4096
#define SLEN 81
void append(FILE *source, FILE *dest);
int main(int argc, char *argv[])
{
    FILE *fa, *fs;
    int files = 0;
    int fct;
    if (argc < 3)
    {
        printf("Использование: %s дополняемый-файл исходный-файл[исходные-файлы]\n",
               argv[0]);
        exit(EXIT_FAILURE);
    }
    if ((fa = fopen(argv[1], "a")) == NULL)
    {
        fprintf(stderr, "Не удается открыть %s\n", argv[1]);
        exit(EXIT_FAILURE);
    }
    if (setvbuf(fa, NULL, _IOFBF, BUFSIZE) != 0)
    {
        fputs("Не удается создать буфер вывода\n", stderr);
        exit(EXIT_FAILURE);
    }
    for (fct = 2; fct < argc; fct++)
    {
        if (strcmp(argv[fct], argv[1]) == 0)
            fputs("Невозможно добавить файл к самому себе\n",stderr);
        else if ((fs = fopen(argv[fct], "r")) == NULL)
            fprintf(stderr, "Не удается открыть %s\n", argv[fct]);
        else
        {
            if (setvbuf(fs, NULL, _IOFBF, BUFSIZE) != 0)
            {
                fputs("Не удается создать буфер вывода\n",stderr);
                continue;
            }
            append(fs, fa);
            if (ferror(fs) != 0)
                fprintf(stderr,"Ошибка при чтении файла %s.\n",
            argv[fct]);
            if (ferror(fa) != 0)
                fprintf(stderr,"Ошибка при записи файла %s.\n",
```

```
            argv[1]);
            fclose(fs);
            files++;
            printf("Файл %s добавлен.\n", argv[fct]);
        }
    }
    printf("Программа завершена. Добавлено файлов: %d.\n", files);
    fclose(fa);
    return 0;
}
void append(FILE *source, FILE *dest)
{
    size_t bytes;
    static char temp[BUFSIZE]; // выделить память один раз
    while ((bytes = fread(temp,sizeof(char),BUFSIZE,source)) > 0)
        fwrite(temp, sizeof (char), bytes, dest);
}
```

13.7.

```
/* Упражнение по программированию 13.7a */
/* В коде предполагается, что конец строки непосредственно предшествует концу файла. */
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
    int ch1, ch2;
    FILE * f1;
    FILE * f2;
    if (argc != 3)
    {
        printf("Использование: %s файл1 файл2\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    if ((f1 = fopen(argv[1], "r")) == NULL)
    {
        printf("Не удается открыть файл %s для ввода\n", argv[1]);
        exit(EXIT_FAILURE);
    }
    if ((f2 = fopen(argv[2], "r")) == NULL)
    {
        printf("Не удается открыть файл %s для ввода\n", argv[2]);
        exit(EXIT_FAILURE);
    }
    ch1 = getc(f1);
    ch2 = getc(f2);
    while (ch1 != EOF || ch2 != EOF)
    {
        while (ch1 != EOF && ch1 != '\n') /* пропустить после достижения EOF */
        {
            putchar(ch1);
            ch1 = getc(f1);
        }
        if (ch1 != EOF)
        {
            putchar('\n');
            ch1 = getc(f1);
        }
```

```
        while (ch2 != EOF && ch2 != '\n') /* пропустить после достижения EOF */
        {
            putchar(ch2);
            ch2 = getc(f2);
        }
        if (ch2 != EOF)
        {
            putchar('\n');
            ch2 = getc(f2);
        }
    }
    if (fclose(f1) != 0)
        printf("Не удается закрыть файл %s\n", argv[1]);
    if (fclose(f2) != 0)
        printf("Не удается закрыть файл %s\n", argv[2]);
    return 0;
}
/* Упражнение по программированию 13.7б */
/* В коде предполагается, что конец строки непосредственно предшествует концу файла. */
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
    int ch1, ch2;
    FILE * f1;
    FILE * f2;
    if (argc != 3)
    {
        printf("Использование: %s файл1 файл2\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    if ((f1 = fopen(argv[1], "r")) == NULL)
    {
        printf("Не удается открыть файл %s для ввода\n", argv[1]);
        exit(EXIT_FAILURE);
    }
    if ((f2 = fopen(argv[2], "r")) == NULL)
    {
        printf("Не удается открыть файл %s для ввода\n", argv[2]);
        exit(EXIT_FAILURE);
    }
    ch1 = getc(f1);
    ch2 = getc(f2);
    while (ch1 != EOF || ch2 != EOF)
    {
        while (ch1 != EOF && ch1 != '\n') /* пропустить после достижения EOF */
        {
            putchar(ch1);
            ch1 = getc(f1);
        }
        if (ch1 != EOF)
        {
            if (ch2 == EOF)
                putchar('\n');
            else
                putchar(' ');
            ch1 = getc(f1);
        }
```

```
        while (ch2 != EOF && ch2 != '\n') /* пропустить после достижения EOF */
        {
           putchar(ch2);
           ch2 = getc(f2);
        }
        if (ch2 != EOF)
        {
           putchar('\n');
           ch2 = getc(f2);
        }
     }
     if (fclose(f1) != 0)
        printf("Не удается закрыть файл %s\n", argv[1]);
     if (fclose(f2) != 0)
        printf("Не удается закрыть файл %s\n", argv[2]);
     return 0;
}
```

13.9.

```
/* Упражнение по программированию 13.9 */
/* Для упрощения подсчета в каждой строке хранится одно число и слово. */
#include <stdio.h>
#include <stdlib.h>
#define MAX 47
int main(void)
{
   FILE *fp;
   char words[MAX];
   int wordct = 0;
   if ((fp = fopen("wordy", "a+")) == NULL)
   {
      fprintf(stderr,"Не удается открыть файл \"wordy\".\n");
      exit(EXIT_FAILURE);
   }
   // определить текущее количество строк
   rewind(fp);
   while (fgets(words, MAX, fp) != NULL)
      wordct++;
   rewind(fp);
   puts("Введите слова для добавления в файл; для завершения");
   puts("введите символ # в начале строки.");
   while ((fscanf(stdin,"%40s", words) == 1) && (words[0] != '#'))
      fprintf(fp, "%3d: %s\n", ++wordct, words);
   puts("Содержимое файла:");
   rewind(fp); // возврат в начало файла
   while (fgets(words, MAX, fp) != NULL) // чтение строки, включая число
      fputs(words, stdout);
   if (fclose(fp) != 0)
      fprintf(stderr,"Ошибка при закрытии файла\n");
   puts("Программа завершена.");
   return 0;
}
```

13.11.

```
/* Упражнение по программированию 13.11 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
#define SLEN 256
const char *errmesg[] = {"Использование: %s строка имя-файла\n",
                         "Не удается открыть файл %s\n" };
int main(int argc, char *argv[])
{
   FILE *fp;
   char line[SLEN];
   if (argc != 3)
   {
      fprintf(stderr, errmesg[0], argv[0]);
      exit(EXIT_FAILURE);
   }
   if ((fp = fopen(argv[2], "r")) == NULL)
   {
      fprintf(stderr, errmesg[1], argv[2]);
      exit(EXIT_FAILURE);
   }
   while (fgets(line, SLEN, fp) != NULL)
   {
      if (strstr(line, argv[1]) != NULL)
         fputs(line, stdout);
   }
   fclose(fp);
   return 0;
}
```

13.12.

Пример входных данных:

```
0 0 9 0 0 0 0 0 0 0 0 0 5 8 9 9 8 5 2 0 0 0 0 0 0 0 0 0 0
0 0 0 0 9 0 0 0 0 0 0 0 5 8 9 9 8 5 5 2 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 5 8 1 9 8 5 4 5 2 0 0 0 0 0 0 0 0
0 0 0 0 9 0 0 0 0 0 0 0 5 8 9 9 8 5 0 4 5 2 0 0 0 0 0 0 0
0 0 9 0 0 0 0 0 0 0 0 0 5 8 9 9 8 5 0 0 4 5 2 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 5 8 9 1 8 5 0 0 0 4 5 2 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 5 8 9 9 8 5 0 0 0 0 4 5 2 0 0 0 0
5 5 5 5 5 5 5 5 5 5 5 5 5 8 9 9 8 5 5 5 5 5 5 5 5 5 5 5 5
8 8 8 8 8 8 8 8 8 8 8 5 8 9 9 8 5 8 8 8 8 8 8 8 8 8 8 8 8
9 9 9 9 0 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 3 9 9 9 9 9 9 9 9
8 8 8 8 8 8 8 8 8 8 8 5 8 9 9 8 5 8 8 8 8 8 8 8 8 8 8 8 8
5 5 5 5 5 5 5 5 5 5 5 5 5 8 9 9 8 5 5 5 5 5 5 5 5 5 5 5 5
0 0 0 0 0 0 0 0 0 0 0 0 5 8 9 9 8 5 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 5 8 9 9 8 5 0 0 0 0 6 6 0 0 0 0 0
0 0 0 0 2 2 0 0 0 0 0 0 5 8 9 9 8 5 0 0 5 6 0 0 6 5 0 0 0
0 0 0 0 3 3 0 0 0 0 0 0 5 8 9 9 8 5 0 5 6 1 1 1 1 6 5 0 0
0 0 0 0 4 4 0 0 0 0 0 0 5 8 9 9 8 5 0 0 5 6 0 0 6 5 0 0 0
0 0 0 0 5 5 0 0 0 0 0 0 5 8 9 9 8 5 0 0 0 0 6 6 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 5 8 9 9 8 5 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 5 8 9 9 8 5 0 0 0 0 0 0 0 0 0 0 0
```

```
/* Упражнение по программированию 13.12 */
#include <stdio.h>
#include <stdlib.h>
#define ROWS 20
#define COLS 30
#define LEVELS 10
const char trans[LEVELS + 1] = " .':~*=&%@";
void MakePic(int data[][COLS], char pic[][COLS], int rows);
void init(char arr[][COLS], char ch);
```

```
int main()
{
    int row, col;
    int picIn[ROWS][COLS];
    char picOut[ROWS][COLS];
    char fileName[81];
    FILE * infile;
    init(picOut, 'S');
    printf("Введите имя файла: ");
    scanf("%80s", fileName);
    if ((infile = fopen(fileName, "r")) == NULL)
    {
        fprintf(stderr, "Не удается открыть файл с данными.\n");
        exit(EXIT_FAILURE);
    }
    for (row = 0; row < ROWS; row++)
        for (col = 0; col < COLS; col++)
            fscanf(infile, "%d", &picIn[row][col]);
    if (ferror(infile))
    {
        fprintf(stderr, "Ошибка при извлечении данных из файла.\n");
        exit(EXIT_FAILURE);
    }
    MakePic(picIn, picOut, ROWS);
    for (row = 0; row < ROWS; row++)
    {
        for (col = 0; col < COLS; col++)
            putchar(picOut[row][col]);
        putchar('\n');
    }
    return 0;
}
void init(char arr[][COLS], char ch)
{
    int r, c;
    for (r = 0; r < ROWS; r++)
        for (c = 0; c < COLS; c++)
            arr[r][c] = ch;
}
void MakePic(int data[][COLS], char pic[][COLS], int rows)
{
    int row, col;
    for (row = 0; row < rows; row++)
        for (col = 0; col < COLS; col++)
            pic[row][col] = trans[data[row][col]];
}
```

Упражнения по программированию из главы 14

14.1.

```
/* pe14-1.c */
#include <stdio.h>
#include <string.h>
#include <ctype.h>
struct month {
    char name[10];
    char abbrev[4];
```

```
    int days;
    int monumb;
};
const struct month months[12] = {
    {"Январь", "Янв", 31, 1},
    {"Февраль", "Фев", 28, 2},
    {"Март", "Мар", 31, 3},
    {"Апрель", "Апр", 30, 4},
    {"Май", "Май", 31, 5},
    {"Июнь", "Июн", 30, 6},
    {"Июль", "Июл", 31, 7},
    {"Август", "Авг", 31, 8},
    {"Сентябрь", "Сен", 30, 9},
    {"Октябрь", "Окт", 31, 10},
    {"Ноябрь", "Ноя", 30, 11},
    {"Декабрь", "Дек", 31, 12}
};
int days(char * m);
int main(void)
{
    char input[20];
    int daytotal;
    printf("Введите название месяца: ");
    while (scanf("%s", input) == 1 && input[0] != 'q')
    {
        daytotal = days(input);
        if (daytotal > 0)
            printf("С начала года прошло %d дней, включая %s.\n", daytotal, input);
        else
            printf("%s не является допустимым вводом.\n", input);
        printf("Введите следующее название месяца (или q для завершения): ");
    }
    puts("Программа завершена.");
    return 0;
}
int days(char * m)
{
    int total = 0;
    int mon_num = 0;
    int i;
    m[0] = toupper(m[0]);
    for (i = 1; m[i] != '\0'; i++)
        m[i] = tolower(m[i]);
    for (i = 0; i < 12; i++)
        if (strcmp(m, months[i].name) == 0)
        {
            mon_num = months[i].monumb;
            break;
        }
    if (mon_num == 0)
        total = -1;
    else
        for (i = 0; i < mon_num; i++)
            total +=months[i].days;
    return total;
}
```

14.3.

```
/* pe14-3.c */
#include <stdio.h>
#include <string.h>
char * s_gets(char * st, int n);
#define MAXTITL 40
#define MAXAUTL 40
#define MAXBKS 100 /* максимальное количество книг */
struct book {      /* установка шаблона book       */
   char title[MAXTITL];
   char author[MAXAUTL];
   float value;
};
void sortt(struct book * pb[], int n);
void sortv(struct book * pb[], int n);
int main(void)
{
   struct book library[MAXBKS];   /* массив структур типа book */
   struct book * pbk[MAXBKS];     /* указатели для сортировки  */
   int count = 0;
   int index;
   printf("Введите название книги.\n");
   printf("Нажмите [enter] в начале строки, чтобы закончить ввод.\n");
   while (count < MAXBKS && s_gets(library[count].title, MAXTITL) != NULL
         && library[count].title[0] != '\0')
   {
      printf("Теперь введите ФИО автора.\n");
      s_gets(library[count].author, MAXAUTL);
      printf("Теперь введите цену.\n");
      scanf("%f", &library[count].value);
      pbk[count] = &library[count];
      count++;
      while (getchar() != '\n')
         continue;   /* очистить входную строку */
      if (count < MAXBKS)
         printf("Введите название следующей книги.\n");
   }
   printf("Каталог ваших книг:\n");
   for (index = 0; index < count; index++)
      printf("%s авторства %s: $%.2f\n", library[index].title,
   library[index].author, library[index].value);
   printf("Каталог ваших книг, отсортированный по названию:\n");
   sortt(pbk, count);
   for (index = 0; index < count; index++)
      printf("%s авторства %s: $%.2f\n", pbk[index]->title,
   pbk[index]->author, pbk[index]->value);
   sortv(pbk, count);
   printf("Каталог ваших книг, отсортированный по цене:\n");
   for (index = 0; index < count; index++)
      printf("%s авторства %s: $%.2f\n", pbk[index]->title,
   pbk[index]->author, pbk[index]->value);
   return 0;
}
void sortt(struct book * pb[], int n)
{
   int top, search;
   struct book * temp;
```

```
    for (top = 0; top < n -1; top++)
        for (search = top + 1; search < n; search++)
            if (strcmp(pb[search]->title, pb[top]->title) < 0)
            {
                temp = pb[search];
                pb[search] = pb[top];
                pb[top] = temp;
            }
}
void sortv(struct book * pb[], int n)
{
    int top, search;
    struct book * temp;
    for (top = 0; top < n - 1; top++)
        for (search = top + 1; search < n; search++)
            if (pb[search]->value < pb[top]->value)
            {
                temp = pb[search];
                pb[search] = pb[top];
                pb[top] = temp;
            }
}
char * s_gets(char * st, int n)
{
    char * ret_val;
    char * find;
    ret_val = fgets(st, n, stdin);
    if (ret_val)
    {
        find = strchr(st, '\n');     // поиск новой строки
        if (find)                    // если адрес не равен NULL,
            *find = '\0';            // поместить туда нулевой символ
        else
            while (getchar() != '\n')
                continue;            // отбросить остаток строки
    }
    return ret_val;
}
```

14.5.

```
/* pe14-5.c */
#include <stdio.h>
#include <string.h>
#define LEN 14
#define CSIZE 4
#define SCORES 3
struct name {
    char first[LEN];
    char last[LEN];
};
struct student {
    struct name person;
    float scores[SCORES];
    float mean;
};
void get_scores(struct student ar[], int lim);
void find_means(struct student ar[], int lim);
```

```
void show_class(const struct student ar[], int lim);
void show_ave(const struct student ar[], int lim);
int main(void)
{
   struct student class[CSIZE] ={
      { "Флип", "Снайд"},
      { "Клэр", "Воянс"},
      { "Бинго", "Хиггс"},
      { "Фойн", "Хантер"}
   };
   get_scores(class, CSIZE);
   find_means(class, CSIZE);
   show_class(class, CSIZE);
   show_ave(class, CSIZE);
   return 0;
}
void get_scores(struct student ar[], int lim)
{
   int i,j;
   for (i = 0; i < lim; i++)
   {
      printf("Введите %d оценок для %s %s:\n", SCORES,
            ar[i].person.first, ar[i].person.last);
      for (j = 0; j < SCORES; j++)
      {
         while (scanf("%f", &ar[i].scores[j]) != 1)
         {
            scanf("%*s");
            puts("Используйте числовой ввод.");
         }
      }
   }
}
void find_means(struct student ar[], int lim)
{
   int i, j;
   float sum;
   for (i = 0; i < lim; i++)
   {
      for (sum = 0, j = 0; j < SCORES; j++)
         sum += ar[i].scores[j];
      ar[i].mean = sum / SCORES;
   }
}
void show_class(const struct student ar[], int lim)
{
   int i, j;
   char wholename[2*LEN];
   for (i = 0; i < lim; i++)
   {
      strcpy(wholename, ar[i].person.first);
      strcat(wholename, " ");
      strcat(wholename, ar[i].person.last);
      printf("%27s: ", wholename);
      for (j = 0; j < SCORES; j++)
         printf("%6.1f ", ar[i].scores[j]);
      printf(" Средняя оценка = %5.2f\n", ar[i].mean);
   }
}
```

```
void show_ave (const struct student ar[], int lim)
{
   int i, j;
   float total;
   printf("\n%27s: ", "СРЕДНИЕ ОЦЕНКИ ПО ЭКЗАМЕНАМ");
   for (j = 0; j < SCORES; j++)
   {
      for (total = 0, i = 0; i < lim; i++)
         total += ar[i].scores[j];
      printf("%6.2f ", total / lim);
   }
   for (total = 0, i = 0; i < lim; i++)
      total += ar[i].mean;
   printf(" Средняя оценка по всем = %5.2f\n", total / lim);
}
```

14.7.

```
/* pe14-7.c */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#define MAXTITL 40
#define MAXAUTL 40
#define MAXBKS 10 /* максимальное количество книг */
#define CONTINUE 0
#define DONE 1
struct book {    /* определение шаблона book    */
   char title[MAXTITL];
   char author[MAXAUTL];
   float value;
};
struct pack {
   struct book book; // разные пространства имен для двух идентификаторов book
   bool delete_me;
};
/*
   Стратегия: вместо того, чтобы перегруппировывать массив структур каждый раз,
   когда происходит удаление, мы объединим структуру с членом данных, который
   указывает, помечен ли элемент для последующего удаления. В конце программы
   будут отображаться и сохраняться только те элементы, которые не помечены как
   подлежащие удалению. Информацию об удалении можно было бы хранить в отдельном
   массиве, но помещение ее в структуру вместе со структурой book позволяет держать
   информацию вместе.
*/
char * s_gets(char * st, int n);
int getlet(const char * s);
int getbook(struct pack * pb);
void update(struct pack * item);
int main(void)
{
   struct pack library[MAXBKS]; /* массив структур */
   int count = 0;
   int deleted = 0;
   int index, filecount, open;
   FILE * pbooks;
   int size = sizeof (struct book);
```

```
if ((pbooks = fopen("book.dat", "r")) != NULL)
{
   while (count < MAXBKS && fread(&library[count], size, 1, pbooks) == 1)
   {
      if (count == 0)
         puts("Текущее содержимое файла book.dat:");
      printf("%s авторства %s: $%.2f\n",library[count].book.title,
      library[count].book.author, library[count].book.value);
      printf("Хотите изменить или удалить эту запись?<y/n> ");
      if (getlet("yn") == 'y')
      {
         printf("Введите c для изменения или d для удаления записи: ");
         if (getlet("cd") == 'd')
         {
            library[count].delete_me = true;
            deleted++;
            puts("Запись помечена для удаления.");
         }
         else
            update(&library[count]);
      }
      count++;
   }
   fclose(pbooks);
}
filecount = count - deleted;
if (count == MAXBKS)
{
   fputs("Файл book.dat заполнен.", stderr);
   exit(EXIT_FAILURE);
}
puts("Введите названия новых книг.");
puts("Нажмите [enter] в начале строки, чтобы закончить ввод.");
open = 0;
while (filecount < MAXBKS)
{
   if (filecount < count)
   {
      while (library[open].delete_me == false)
         open++;
      if (getbook(&library[open]) == DONE)
         break;
   }
   else if (getbook(&library[filecount]) == DONE)
      break;
   filecount++;
   if (filecount < MAXBKS)
      puts("Введите название следующей книги.");
}
puts("Каталог ваших книг:");
for (index = 0; index < filecount; index++)
   if (library[index].delete_me == false)
      printf("%s авторства %s: $%.2f\n",library[index].book.title,
               library[index].book.author, library[index].book.value);
if ((pbooks = fopen("book.dat", "w")) == NULL)
{
   fputs("Не удается открыть файл book.dat для вывода.\n",stderr);
   exit(EXIT_FAILURE);
}
```

```
    for (index = 0; index < filecount; index++)
        if (library[index].delete_me == false)
            fwrite(&(library[index].book), size, 1, pbooks);
    fclose(pbooks);
    puts("Программа завершена.");
    return 0;
}
int getlet(const char * s)
{
    char c;
    c = getchar();
    while (strchr(s, c) == NULL)
    {
        printf ("Введите символ из списка %s\n", s);
        while( getchar() != '\n')
        continue;
        c = getchar();
    }
    while (getchar() != '\n')
        continue;
    return c;
}
int getbook(struct pack * pb)
{
    int status = CONTINUE;
    if (s_gets(pb->book.title, MAXTITL) == NULL || pb->book.title[0] == '\0')
        status = DONE;
    else
    {
        printf ("Теперь введите имя автора: ");
        s_gets (pb->book.author, MAXAUTL);
        printf ("Теперь введите цену книги: ");
        while (scanf("%f", &pb->book.value ) != 1)
        {
            puts("Используйте числовой ввод.");
            scanf("%*s");
        }
        while (getchar() != '\n')
            continue; /* очистить входную строку */
        pb->delete_me = false;
    }
    return status;
}
void update(struct pack * item)
{
    struct book copy;
    char c;
    copy = item->book;
    puts("Введите букву, обозначающую желаемое действие:");
    puts("t) изменить название  a) изменить автора");
    puts("v) изменить цену      s) выйти с сохранением изменений");
    puts("q) выйти без сохранения изменений");
    while ( (c = getlet("tavsq")) != 's' && c != 'q')
    {
        switch ( c )
        {
```

```
            case 't' : puts("Введите новое название: ");
                       s_gets (copy.title, MAXTITL);
                       break;
            case 'a' : puts("Введите ФИО нового автора: ");
                       s_gets (copy.author, MAXAUTL);
                       break;
            case 'v' : puts("Введите новую цену: ");
                       while (scanf("%f", &copy.value) != 1)
                       {
                          puts ("Введите числовое значение: ");
                          scanf("%*s");
                       }
                       while( getchar() != '\n')
                          continue;
                       break;
         }
         puts("t) изменить название  a) изменить автора");
         puts("v) изменить цену      s) выйти с сохранением изменений");
         puts("q) выйти без сохранения изменений");
      }
      if (c == 's')
         item->book = copy;
   }
   char * s_gets(char * st, int n)
   {
      char * ret_val;
      char * find;
      ret_val = fgets(st, n, stdin);
      if (ret_val)
      {
         find = strchr(st, '\n');  // поиск новой строки
         if (find)                 // если адрес не равен NULL,
            *find = '\0';          // поместить туда нулевой символ
         else
            while (getchar() != '\n')
               continue;           // отбросить остаток строки
      }
      return ret_val;
   }
```

14.8.

```
/* pe14-8.c */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define LEN 14
#define SEATS 12
#define EMPTY 0
#define TAKEN 1
#define CONTINUE 1
#define DONE 0
struct planestats {
   int seat_id;
   int status;
   char last[LEN];
   char first[LEN];
};
```

```
int getmenu(void);
int getlet(const char *);
int openings(const struct planestats [], int);
void show_empties(const struct planestats [], int);
void list_assign(struct planestats *[], int);
void assign_seat(struct planestats [], int);
void delete_seat(struct planestats [], int);
void show_seats(const struct planestats [], int);
void sort(struct planestats *[], int);
void makelist(const struct planestats [], char *, int);
char * s_gets(char * st, int n);
int main(void)
{
   struct planestats plane_1[SEATS], *ps[SEATS];
   int choice;
   int i;
   FILE *fp;
   size_t size = sizeof(struct planestats);
   for (i = 0; i < SEATS; i++)
      ps[i] = &plane_1[i];
   if ((fp = fopen("air.dat", "rb")) == NULL )
      for (i = 0; i < SEATS; i++)
      {
         plane_1[i].status = EMPTY;
         plane_1[i].seat_id = i + 1;
      }
   else
   {
      fread(plane_1, size, SEATS, fp);
      fclose(fp);
   }
   while ( (choice = getmenu() ) != 'q')
   {
      switch (choice)
      {
         case 'o' : printf("Количество свободных мест: %d.\n",
                   openings(plane_1, SEATS));
                   break;
         case 'e' : show_empties(plane_1, SEATS);
                   break;
         case 'l' : list_assign(ps, SEATS);
                   break;
         case 'a' : assign_seat(plane_1, SEATS);
                   break;
         case 'd' : delete_seat(plane_1, SEATS);
                   break;
         default :  puts("Проблема в операторе switch.");
                   break;
      }
   }
   if((fp = fopen("air.dat", "wb")) == NULL )
      puts("Не удается сохранить данные в файл.");
   else
   {
      fwrite(plane_1, size, SEATS, fp);
      fclose(fp);
   }
```

```
    puts("Программа завершена.");
    return 0;
}
#define CHOICES 6
int getmenu(void)
{
    const char *descript[CHOICES] = {
        "Показать количество свободных мест",
        "Показать список свободных мест",
        "Показать список забронированных мест в алфавитном порядке",
        "Забронировать место для пассажира",
        "Снять броню с места",
        "Выйти из программы"
    };
    const char labels[CHOICES + 1] = "oeladq";
    int i;
    puts("Для выбора функции введите ее буквенную метку:");
    for (i = 0; i < CHOICES; i++)
        printf("%c) %s\n", labels[i], descript[i]);
    return getlet(labels);
}
int getlet(const char * s)
{
    char c;
    c = getchar();
    while (strchr(s, c) == NULL)
    {
        printf ("Введите символ из списка %s\n", s);
        while (getchar() != '\n')
            continue;
        c = getchar();
    }
    while (getchar() != '\n')
        continue;
    return c;
}
int openings(const struct planestats pl[], int n)
{
    int count = 0;
    int seat;
    for (seat = 0; seat < n; seat++)
        if (pl[seat].status == EMPTY)
            count++;
    return count;
}
void show_empties(const struct planestats pl[], int n)
{
    char seating[3* SEATS];
    if (openings(pl,n) == 0)
        puts("Все места забронированы.");
    else
    {
        puts("Свободны следующие места:");
        makelist(pl, seating, EMPTY);
        puts(seating);
    }
}
```

```
void makelist(const struct planestats pl[], char * str, int kind)
{
   int seat;
   char temp[LEN];
   str[0] = '\0';
   for (seat = 0; seat < SEATS; seat++)
      if (pl[seat].status == kind)
      {
         sprintf(temp," %d", pl[seat].seat_id);
         strcat(str, temp);
      }
}
void list_assign(struct planestats *ps[], int n)
{
   int i;
   if (openings(*ps, n) == SEATS)
      puts("Все места свободны.");
   else
   {
      sort(ps, n);
      for(i = 0; i < SEATS; i++)
         if ( ps[i]->status == TAKEN )
            printf ("Место %d: %s, %s\n",
      ps[i]->seat_id, ps[i]->last, ps[i]->first);
   }
}
void assign_seat(struct planestats pl[], int n)
{
   char list[3 * SEATS];
   int seat, loop;
   if (openings(pl,n) == 0)
      puts("Все места забронированы.");
   else
   {
      makelist(pl,list, EMPTY);
      puts("Какое место вас интересует? Выберите из следующего списка:");
      puts(list);
      do
      {
         while( scanf("%d", &seat) != 1)
         {
            scanf("%*s");
            puts("Введите число из этого списка:");
            puts(list);
         }
         if (seat < 1 || seat > SEATS ||
            pl[seat-1].status == TAKEN)
         {
            puts("Введите число из этого списка:");
            puts(list);
            loop = CONTINUE;
         }
         else
            loop = DONE;
      } while (loop == CONTINUE);
      while (getchar() != '\n')
         continue;
```

```
        puts("Введите имя:");
        s_gets(pl[seat - 1].first, LEN);
        puts("Введите фамилию:");
        s_gets(pl[seat - 1].last, LEN);
        printf("Для %s %s бронируется место %d.\n",
        pl[seat - 1].first, pl[seat - 1].last, seat);
        puts("Введите a, чтобы принять броню, или c, чтобы отменить бронирование.");
        if (getlet("ac") == 'a')
        {
            pl[seat - 1].status = TAKEN;
            puts("Место для пассажира забронировано.");
        }
        else
            puts("Место для пассажира не забронировано.");
    }
}
void delete_seat(struct planestats pl[], int n)
{
    int seat, loop;
    char list[3 * SEATS];
    if (openings(pl, n) == SEATS)
        puts("Все места уже свободны.");
    else
    {
        show_seats(pl, n);
        makelist(pl, list, TAKEN);
        puts("Введите номер места, с которого должна быть снята броня:");
        do
        {
            while( scanf("%d", &seat) != 1)
            {
                scanf("%*s");
                puts("Введите число из этого списка:");
                puts(list);
            }
            if (seat < 1 || seat > SEATS ||
                pl[seat-1].status == EMPTY)
            {
                puts("Введите число из этого списка:");
                puts(list);
                loop = CONTINUE;
            }
            else
                loop = DONE;
        } while (loop == CONTINUE);
        while (getchar() != '\n')
            continue;
        printf("Для %s %s снимается броня с места %d.\n",
        pl[seat - 1].first, pl[seat - 1].last, seat);
        puts("Введите d, чтобы снять броню, или a, чтобы отменить снятие.");
        if (getlet("da") == 'd')
        {
            pl[seat - 1].status = EMPTY;
            puts("Броня для пассажира снята.");
        }
        else
            puts("Броня для пассажира оставлена.");
    }
}
```

```
void show_seats(const struct planestats pl[], int n)
{
   int i;
   puts("Места, которые в настоящее время забронированы:");
   for (i = 0; i < SEATS; i++)
   if (pl[i].status == TAKEN)
      printf("Место %d: %s, %s\n", pl[i].seat_id,
   pl[i].last, pl[i].first);
}

void sort(struct planestats *array[], int limit)
{
   int top, search;
   struct planestats * temp;
   for (top = 0; top < limit -1; top++)
      for (search = top + 1; search < limit; search++)
         if (strcmp(array[search]->last, array[top]->last) < 0)
         {
            temp = array[search];
            array[search] = array[top];
            array[top] = temp;
         }
}

char * s_gets(char * st, int n)
{
   char * ret_val;
   char * find;
   ret_val = fgets(st, n, stdin);
   if (ret_val)
   {
      find = strchr(st, '\n');  // поиск новой строки
      if (find)                 // если адрес не равен NULL,
         *find = '\0';          // поместить туда нулевой символ
      else
         while (getchar() != '\n')
            continue;          // отбросить остаток строки
   }
   return ret_val;
}
```

14.10.

```
/* pe14-10.c */
/* Сложная часть связана с объявлением массива указателей на функции. */
#include <stdio.h>
#include <math.h>       // для sqrt()
double twice(double x);
double half(double x);
double thrice(double x);
void showmenu(void);
#define NUM 4
int main(void)
{
   double (*pf[NUM])(double) = {twice, half, thrice, sqrt};
   double val;
   double ans;
   int sel;
   printf("Введите число (отрицательное приводит к завершению): ");
```

```
    while (scanf("%lf", &val) && val >= 0)
    {
        showmenu();
        while (scanf("%d", &sel) && sel >= 0 && sel <= 3)
        {
            ans = (*pf[sel])(val); // первая форма записи
            printf("ответ = %f\n", ans);
            ans = pf[sel](val);    // альтернативная форма записи
            printf("чтобы повторить, ответ = %f\n", ans);
            showmenu();
        }
        printf("Введите следующее число (отрицательное приводит к завершению): ");
    }
    puts("Программа завершена.");
    return 0;
}

void showmenu(void)
{
    puts("Введите один из следующих вариантов:");
    puts("0) удвоить значение  1) разделить значение пополам");
    puts("2) утроить значение  3) взять квадратный корень значения");
    puts("4) ввести следующее число");
}

double twice(double x) {return 2.0 * x;}
double half(double x) {return x / 2.0;}
double thrice(double x) {return 3.0 * x;}
```

Упражнения по программированию из главы 15

15.1.

```
/* pe15-1.c */
#include <stdio.h>
#include <stdbool.h>  // C99 -- в противном случае используйте int
#include <limits.h>   // для CHAR_BIT
#include <string.h>   // для strchr()
int bstr_to_dec(const char * str);
bool check_val(const char * str);
char * s_gets(char * st, int n);
int main(void)
{
    const size_t SLEN = CHAR_BIT * sizeof(int) + 1;
    char value[SLEN];
    printf("Введите двоичное число, содержащее до %zu цифр: ", SLEN - 1);
    while (s_gets(value, SLEN) && value[0] != '\0')
    {
        if (!check_val(value))
            puts("Двоичное число может содержать только цифры 0 и 1.");
        else
            printf("%s является %d\n", value, bstr_to_dec(value));
        puts("Введите следующее число (или пустую строку для завершения):");
    }
    puts("Программа завершена.");
    return 0;
}
```

```
int bstr_to_dec(const char * str)
{
   int val = 0;
   while (*str != '\0')
   val = 2 * val + (*str++ - '0');
   return val;
}
bool check_val(const char * str)
{
   bool valid = true;
   while (valid && *str != '\0')
   {
      if (*str != '0' && *str != '1')
         valid = false;
      ++str;
   }
   return valid;
}
char * s_gets(char * st, int n)
{
   char * ret_val;
   char * find;
   ret_val = fgets(st, n, stdin);
   if (ret_val)
   {
      find = strchr(st, '\n'); // поиск новой строки
      if (find)                // если адрес не равен NULL,
         *find = '\0';         // поместить туда нулевой символ
      else
         while (getchar() != '\n')
            continue;          // отбросить остаток строки
   }
   return ret_val;
}
```

15.2.

```
/* pe15-2.c */
#include <stdio.h>
#include <stdlib.h>
int bstr_to_dec(const char * str);
char * itobs(int, char *);
int main(int argc, char * argv[])
{
   int v1;
   int v2;
   char bstr[8* sizeof (int) + 1];
   if (argc != 3)
   {
      fprintf(stderr, "Использование: %s двоичное-число-1 двоичное-число-2\n",
              argv[0]);
      exit(EXIT_FAILURE);
   }
   v1 = bstr_to_dec(argv[1]);
   v2 = bstr_to_dec(argv[2]);
   printf("~%s = %s\n", argv[1], itobs(~v1, bstr));
   printf("~%s = %s\n", argv[2], itobs(~v2, bstr));
   printf("%s & %s= %s\n", argv[1], argv[2], itobs(v1 & v2, bstr));
   printf("%s | %s= %s\n", argv[1], argv[2], itobs(v1 | v2, bstr));
```

```
    printf("%s ^ %s= %s\n", argv[1], argv[2], itobs(v1 ^ v2, bstr));
    puts("Программа завершена.");
    return 0;
}

int bstr_to_dec(const char * str)
{
    int val = 0;
    while (*str != '\0')
        val = 2 * val + (*str++ - '0');
    return val;
}

char * itobs(int n, char * ps)
{
    int i;
    static int size = 8 * sizeof(int);
    for (i = size - 1; i >= 0; i--, n >>= 1)
        ps[i] = (01 & n) + '0';
    ps[size] = '\0';
    return ps;
}
```

15.3.

```
/* pe15-3.c */
#include <stdio.h>
#include <limits.h>
char * itobs(int, char *);
int onbits(int);
int main(int argc, char * argv[])
{
    int val;
    char bstr[CHAR_BIT * sizeof(int) + 1];
    printf("Введите целое число (или q для завершения): ");
    while (scanf("%d", &val))
    {
        printf ("%d (%s) имеет %d установленных бит(ов).\n", val, itobs(val, bstr),
                onbits(val));
        printf("Введите следующее целое число: ");
    }
    puts("Программа завершена.");
    return 0;
}

char * itobs(int n, char * ps)
{
    int i;
    static int size = CHAR_BIT * sizeof(int);
    for (i = size - 1; i >= 0; i--, n >>= 1)
        ps[i] = (01 & n) + '0';
    ps[size] = '\0';
    return ps;
}

int onbits(int n)
{
    static const int size = CHAR_BIT * sizeof(int);
    int ct = 0;
    int i;
    for (i = 0; i < size; i++, n >>= 1)
```

```
        if ((1 & n) == 1)
            ct++;
    return ct;
}
```

15.5.

```
/* pe15-5.c */
#include <stdio.h>
#include <limits.h>
unsigned int rotate_l(unsigned int, unsigned int);
char * itobs(int, char *);
int main(void)
{
    unsigned int val;
    unsigned int rot;
    unsigned int places;
    char bstr1[CHAR_BIT * sizeof (int) + 1];
    char bstr2[CHAR_BIT * sizeof (int) + 1];
    printf("Введите целое число (или q для завершения): ");
    while (scanf("%ud", &val))
    {
        printf("Введите количество битов, на которое нужно выполнить циклический
сдвиг: \n");
        if (scanf("%ul", &places) != 1)
            break;
        rot = rotate_l(val, places);
        itobs(val, bstr1);
        itobs(rot, bstr2);
        printf ("%u после циклического сдвига равно %u.\n", val, rot );
        printf("%s после циклического сдвига равно %s.\n", bstr1, bstr2);
        printf("Введите следующее целое число: ");
    }
    puts("Программа завершена.");
    return 0;
}

unsigned int rotate_l(unsigned int n, unsigned int b)
{
    static const int size = CHAR_BIT * sizeof(int);
    unsigned int overflow;
    b %= size;                      /* сохранить b допустимым значением*/
    overflow = n >> (size - b);     /* сохранить сдвинутые биты        */
    return (n << b) | overflow;
}

char * itobs(int n, char * ps)
{
    int i;
    const static int size = CHAR_BIT * sizeof(int);
    for (i = size - 1; i >= 0; i--, n >>= 1)
        ps[i] = (01 & n) + '0';
    ps[size] = '\0';
    return ps;
}
```

15.7.

```
// pe15-7.c
#include <stdio.h>
#include <string.h>
```

```
#include <ctype.h>
#define ID_MASK 0xFF
#define SIZE_MASK 0x7F00
#define LEFT 0x00000
#define CENTER 0x08000
#define RIGHT 0x10000
#define ALIGN_MASK 0x18000
#define REGULAR 0x00000
#define BOLD 0x20000
#define ITALIC 0x40000
#define UNDERLINE 0x80000
#define STYLE_MASK 0xE0000
#define SIZE_SHIFT 8
typedef unsigned long font;
char do_menu(font * f);
char get_choice(const char *);
void show_menu(void);
void show_font(font f);
void eatline(void);
void get_id(font * f);
void get_size(font * f);
void get_align(font * f);
int main(void)
{
   font sample = 1 | (12 <<SIZE_SHIFT) | LEFT | ITALIC;
   while (do_menu(&sample) != 'q')
      continue;
   puts("Программа завершена.");
   return 0;
}
char do_menu(font * f)
{
   char response;
   show_font(*f);
   show_menu();
   response = get_choice("fsabiuq");
   switch(response)
   {
      case 'f' : get_id(f); break;
      case 's' : get_size(f); break;
      case 'a' : get_align(f); break;
      case 'b' : *f ^= BOLD; break;
      case 'i' : *f ^= ITALIC; break;
      case 'u' : *f ^= UNDERLINE; break;
      case 'q' : break;
      default : fprintf(stderr, "проблема с меню\n");
   }
   return response;
}
char get_choice(const char * str)
{
   char ch;
   ch = getchar();
   ch = tolower(ch);
   eatline();
   while (strchr(str, ch) == NULL)
   {
```

```
        printf("Введите один из следующих вариантов: %s\n", str);
        ch = tolower(getchar());
        eatline();
    }
    return ch;
}
void eatline(void)
{
    while (getchar() != '\n')
        continue;
}
void show_menu(void)
{
    puts("f) изменить шрифт        s) изменить размер    a) изменить
выравнивание");
    puts("b) переключить полужирный  i) переключить курсив  u) переключить
подчеркнутый");
    puts("q) завершить");
}
void show_font(font f)
{
    printf("\n%4s %6s %12s %5s %5s %5s\n",
          "ИД", "РАЗМЕР", "ВЫРАВНИВАНИЕ", "Ж", "К", "П");
    printf("%4lu %4lu", f & ID_MASK, (f & SIZE_MASK) >> SIZE_SHIFT);
    switch(f & ALIGN_MASK)
    {
        case LEFT : printf("%10s", "влево"); break;
        case RIGHT : printf("%10s", "вправо"); break;
        case CENTER : printf("%10s", "по центру"); break;
        default : printf("%10s", "неизвестно"); break;
    }
    printf("%8s %5s %5s\n\n", (f & BOLD) == BOLD? "вкл." : "откл.",
          (f & ITALIC) == ITALIC ? "вкл." : "откл.",
          (f & UNDERLINE) == UNDERLINE ? "вкл." : "откл.");
}
void get_id(font * f)
{
    int id;
    printf("Введите идентификатор шрифта (0-255): ");
    scanf("%d", &id);
    id = id & ID_MASK;
    *f |= id;
    eatline();
}
void get_size(font * f)
{
    int size;
    printf("Введите размер шрифта (0-127): ");
    scanf("%d", &size);
    *f |= (size << SIZE_SHIFT) & SIZE_MASK;
    eatline();
}
void get_align(font * f)
{
    puts("Выберите выравнивание:");
    puts("l) влево  c) по центру  r) вправо");
    switch (get_choice("lcr"))
    {
```

```
        case 'l' : *f &= ~ALIGN_MASK; *f |= LEFT; break;
        case 'c' : *f &= ~ALIGN_MASK; *f |= CENTER; break;
        case 'r' : *f &= ~ALIGN_MASK; *f |= RIGHT; break;
        default : fprintf(stderr, "проблема с выравниванием\n");
    }
}
```

Упражнения по программированию из главы 16

16.2.

```
/* pe16-2.c */
#include <stdio.h>
#define HMEAN(X,Y) (2.0 * (X) *(Y) / ((X) + (Y)))
int main(void)
{
    double x, y, ans;
    puts("Введите пару чисел (или q для завершения): ");
    while (scanf("%lf %lf", &x, &y) == 2)
    {
        ans = HMEAN(x,y);
        printf("%g = гармоническое среднее для %g %g.\n", ans, x, y);
        // посмотреть, работает ли макрос с арифметическими выражениями
        ans = HMEAN(x + y, x * y);
        printf("%g = гармоническое среднее для %g %g.\n", ans, x + y, x * y);
        puts("Введите пару чисел (или q для завершения): ");
    }
    puts("Программа завершена.");
    return 0;
}
```

16.3.

```
/* pe16-3.c */
#include <stdio.h>
#include <math.h>
struct polar {
double r;
double theta; /* угол в градусах */
};
struct rect {
    double x;
    double y;
};
struct rect p_to_r(const struct polar * ppol);
int main(void)
{
    struct polar input;
    struct rect answer;
    printf("Введите модуль и угол в градусах: ");
    while (scanf("%lf %lf", &input.r, &input.theta) == 2)
    {
        answer = p_to_r(&input);
        printf("полярные координаты: %g %f\n",input.r, input.theta);
        printf("прямоугольные координаты: %g %g\n",answer.x, answer.y);
        printf("Введите модуль и угол в градусах (или q для завершения): ");
    }
    puts("Программа завершена.");
    return 0;
}
```

```
struct rect p_to_r(const struct polar * ppol)
{
   static const double deg_rad = 3.141592654 / 180.0;
   struct rect res;
   double ang = deg_rad * ppol->theta; /* преобразование градусов в радианы */
   res.x = ppol->r * sin(ang);
   res.y = ppol->r * cos(ang);
   return res;
}
```

16.5.

```
/* pe16-5.c */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void random_pick(int ar[], int arsize, int picks);
#define SPOTS 51
#define PICKS 6
int main(void)
{
   int lotto[SPOTS];
   int i;
   char ch;
   for (i = 0; i < SPOTS; i++)
      lotto[i] = i + 1;
   do {
      random_pick(lotto, SPOTS, PICKS);
      printf ("Еще раз? <y/n> ");
      ch = getchar();
      while (getchar() != '\n')
         continue;
   } while (ch == 'y' || ch == 'Y');
   puts("Программа завершена.");
   return 0;
}
void random_pick(int ar[], int arsize, int picks)
{
   int i, index, temp;
   srand((unsigned int) time(0));
   if (picks > arsize)
   {
      fputs("Количество выбранных элементов превышает размер массива\n", stderr);
      fputs("Количество выбранных элементов делается равным размеру массива\n", stderr);
      picks = arsize;
   }
   for (i = 0; i < picks; i++)
   {
      index = rand() % (arsize - 1); /* выбрать случайный элемент */
      temp = ar[index];
      printf ("%2d ", temp);         /* отобразить его */
      if (i % 20 == 19)
         putchar('\n');
      ar[index] = ar[arsize - 1];    /* поменять его с последним элементом */
      ar[arsize - 1] = temp;
      arsize--;                      /* исключить последний элемент из поиска */
   }
   if (i % 20 != 0)
      putchar('\n');
}
```

16.7.

```
// pe16-7.c.-- использование функции с переменным числом аргументов
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
void show_array(const double ar[], int n);
double * new_d_array(int n, ...);
int main()
{
   double * p1;
   double * p2;
   p1 = new_d_array(5, 1.2, 2.3, 3.4, 4.5, 5.6);
   p2 = new_d_array(4, 100.0, 20.00, 8.08, -1890.0);
   show_array(p1, 5);
   show_array(p2, 4);
   free(p1);
   free(p2);
   return 0;
}

void show_array(const double ar[], int n)
{
   int i;
   for (i = 0; i < n; i++)
      printf("%g ", ar[i]);
   putchar('\n');
}

double * new_d_array(int n, ...)
{
   va_list ap;
   int i;
   double * pt;
   va_start(ap, n);
   pt = (double *) malloc(n * sizeof(double));
   for (i = 0; i< n; i++)
      pt[i] = va_arg(ap, double);
   va_end(ap);
   return pt;
}
```

Упражнения по программированию из главы 17

17.1a.

```
/* pe17-1a.c -- рекурсивное решение */
#include <stdio.h>
#include <stdlib.h> /* содержит прототип функции malloc()   */
#include <string.h> /* содержит прототип функции strcpy()   */
#define TSIZE 45   /* размер массива для хранения названия */
struct film {
   char title[TSIZE];
   int rating;
   struct film * next; /* указывает на следующую структуру в списке */
};
char * s_gets(char * st, int n);
void show_rec(const struct film * pf); /* рекурсивная функция */
int main(void)
{
```

```
    struct film * head = NULL;
    struct film * prev, * current;
    char input[TSIZE];
    puts("Введите название первого фильма:");
    while (s_gets(input, TSIZE) != NULL && input[0] != '\0')
    {
        current = (struct film *) malloc(sizeof(struct film));
        if (head == NULL)         /* первая структура     */
            head = current;
        else                      /* последующие структуры */
            prev->next = current;
        current->next = NULL;
        strcpy(current->title, input);
        puts("Введите свое значение рейтинга <0-10>:");
        scanf("%d", &current->rating);
        while(getchar() != '\n')
            continue;
        puts("Введите название следующего фильма (или пустую строку для прекращения
ввода):");
        prev = current;
    }
    if (head == NULL)
        printf("Данные не введены.");
    else
        printf ("Список фильмов:\n");
    current = head;
    while (current != NULL)
    {
        printf("Фильм: %s  Рейтинг: %d\n", current->title, current->rating);
        current = current->next;
    }
    if (head != NULL)
    {
        printf("\nСписок фильмов в обратном порядке:\n");
        show_rec(head);
    }
    printf("Программа завершена.\n");
    return 0;
}
char * s_gets(char * st, int n)
{
    char * ret_val;
    char * find;
    ret_val = fgets(st, n, stdin);
    if (ret_val)
    {
        find = strchr(st, '\n');  // поиск новой строки
        if (find)                 // если адрес не равен NULL,
            *find = '\0';         // поместить туда нулевой символ
        else
            while (getchar() != '\n')
                continue;         // отбросить остаток строки
    }
    return ret_val;
}
void show_rec(const struct film * pf)
{
```

```
    if (pf->next != NULL)
        show_rec(pf->next);
    printf("Фильм: %s  Рейтинг: %d\n", pf->title, pf->rating);
}
```

17.16.

```
/* pe17-1b.c -- решение с двухсвязным списком */
#include <stdio.h>
#include <stdlib.h>        /* содержит прототип функции malloc()   */
#include <string.h>        /* содержит прототип функции strcpy()   */
#define TSIZE 45           /* размер массива для хранения названия */
struct film {
    char title[TSIZE];
    int rating;
    struct film * next;    /* указывает на следующую структуру в списке  */
    struct film * prev;    /* указывает на предыдущую структуру в списке */
};
char * s_gets(char * st, int n);
int main(void)
{
    struct film * head = NULL;
    struct film * prev, * current;
    char input[TSIZE];
    puts("Введите название первого фильма:");
    while (s_gets(input, TSIZE) != NULL && input[0] != '\0')
    {
        current = (struct film *) malloc(sizeof(struct film));
        if (head == NULL)    /* первая структура     */
        {
            head = current;
            head->prev = NULL;
        }
        else                 /* последующие структуры */
        {
            prev->next = current;
            current->prev = prev;
        }
        current->next = NULL;
        strcpy(current->title, input);
        puts("Введите свое значение рейтинга <0-10>:");
        scanf("%d", &current->rating);
        while(getchar() != '\n')
            continue;
        puts("Введите название следующего фильма (или пустую строку для прекращения
ввода):");
        prev = current;
    }
    if (head == NULL)
        printf("Данные не введены.");
    else
        printf ("Список фильмов:\n");
    current = head;
    while (current != NULL)
    {
        printf("Фильм: %s  Рейтинг: %d\n", current->title, current->rating);
        prev = current;
        current = current->next;
    }
```

```
    if (head != NULL)
    {
       printf("\nСписок фильмов в обратном порядке:\n");
       current = prev;
       while (current != NULL)
       {
          printf("Фильм: %s  Рейтинг: %d\n", current->title, current->rating);
          current = current->prev;
       }
    }
    printf("Программа завершена.\n");
    return 0;
}
char * s_gets(char * st, int n)
{
    char * ret_val;
    char * find;
    ret_val = fgets(st, n, stdin);
    if (ret_val)
    {
       find = strchr(st, '\n');  // поиск новой строки
       if (find)                 // если адрес не равен NULL,
          *find = '\0';          // поместить туда нулевой символ
       else
          while (getchar() != '\n')
             continue;          // отбросить остаток строки
    }
    return ret_val;
}
```

17.3.

```
/* list17-3.h -- заголовочный файл для простого спискового типа */
#ifndef LIST_H_
#define LIST_H_
#include <stdbool.h> /* C99 -- иначе определите bool с помощью enum */
/* объявления, специфичные для программы */
#define TSIZE 45 /* размер массива для хранения названия */
struct film
{
    char title[TSIZE];
    int rating;
};
/* объявления общих типов */
typedef struct film Item;
typedef struct node
{
    Item item;
    struct node * next;
} Node;
#define MAXSIZE 100
typedef struct list
{
    Item entries[MAXSIZE];   /* массив элементов     */
    int items;               /* количество элементов */
} List;
/* прототипы функций                                          */
/* операция:    инициализация списка                          */
```

```
/* предусловия:  plist указывает на список                          */
/* постусловия:  список инициализирован пустым содержимым           */
void InitializeList(List * plist);

/* операция:     определение, является ли список пустым             */
/* предусловия:  plist указывает на инициализированный список       */
/* постусловия:  функция возвращает значение True, если список      */
/*               пуст, и False в противном случае                   */
bool ListIsEmpty(const List * plist);

/* операция:    определение, является ли список полным              */
/* предусловия:  plist указывает на инициализированный список       */
/* постусловия:  функция возвращает значение True, если список      */
/*               полон, и False в противном случае                  */
bool ListIsFull(const List * plist);

/* операция:     определение количества элементов в списке          */
/* предусловия:  plist указывает на инициализированный список       */
/* постусловия:  функция возвращает число элементов в списке        */
unsigned int ListItemCount(const List * plist);

/* операция:     добавление элемента в конец списка                 */
/* предусловия:  item – элемент, добавляемый в список               */
/*               plist указывает на инициализированный список       */
/* постусловия:  если возможно, функция добавляет элемент в         */
/*               конец списка и возвращает значение True;           */
/*               в противном случае возвращается значение False     */
bool AddItem(Item item, List * plist);

/* операция:     применение функции к каждому элементу списка       */
/* предусловия:  plist указывает на инициализированный список       */
/*               pfun указывает на функцию, которая принимает       */
/*               аргумент Item и не имеет возвращаемого значения    */
/* постусловия:  функция, указанная pfun, выполняется один раз      */
/*               для каждого элемента в списке                      */
void Traverse (const List * plist, void (* pfun)(Item item) );

/* операция:     освобождение выделенной памяти, если она есть      */
/* предусловия:  plist указывает на инициализированный список       */
/* постусловия:  любая память, выделенная для списка,               */
/*               освобождается, и список устанавливается            */
/*               в пустое состояние                                 */
void EmptyTheList(List * plist);
#endif
```

```
/* pe17-3a.c -- копия films3.c */
/* компилировать вместе с pe17-3b.c */
#include <stdio.h>
#include <stdlib.h>      /* прототип для exit()    */
#include "list17-3.h"    /* определяет List, Item  */
void showmovies(Item item);
char * s_gets(char * st, int n);
int main(void)
{
   List movies;
   Item temp;
   /* инициализация */
   InitializeList(&movies);
   if (ListIsFull(&movies))
   {
      fprintf(stderr,"Доступная память отсутствует! Программа завершена.\n");
      exit(1);
   }
```

```
    /* сбор и сохранение информации */
    puts("Введите название первого фильма:");
    while (s_gets(temp.title, TSIZE) != NULL && temp.title[0] != '\0')
    {
       puts("Введите свое значение рейтинга <0-10>:");
       scanf("%d", &temp.rating);
       while(getchar() != '\n')
          continue;
       if (AddItem(temp, &movies)==false)
       {
          fprintf(stderr, "Проблема с выделением памяти\n");
          break;
       }
       if (ListIsFull(&movies))
       {
          puts("Список полон.");
          break;
       }
       puts("Введите название следующего фильма (или пустую строку для прекращения
ввода):");
    }
    /* отображение */
    if (ListIsEmpty(&movies))
       printf("Данные не введены.");
    else
    {
       printf("Список фильмов:\n");
       Traverse(&movies, showmovies);
    }
    printf("Вы ввели %d фильмов.\n", ListItemCount(&movies));
    /* очистка */
    EmptyTheList(&movies);
    printf("Программа завершена.\n");
    return 0;
}

void showmovies(Item item)
{
    printf("Фильм: %s  Рейтинг: %d\n", item.title,
    item.rating);
}

char * s_gets(char * st, int n)
{
    char * ret_val;
    char * find;
    ret_val = fgets(st, n, stdin);
    if (ret_val)
    {
       find = strchr(st, '\n'); // поиск новой строки
       if (find)                // если адрес не равен NULL,
          *find = '\0';         // поместить туда нулевой символ
       else
          while (getchar() != '\n')
             continue;          // отбросить остаток строки
    }
    return ret_val;
}
```

```
/* pe17-3b.c: переделанный list.c -- функции для поддержки операций со списком */
#include <stdio.h>
#include <stdlib.h>
#include "list17-3.h"
/* функции интерфейса */
/* устанавливает список в пустое состояние */
void InitializeList(List * plist)
{
   plist->items = 0;
}
/* возвращает true, если список пуст */
bool ListIsEmpty(const List * plist)
{
   if (plist->items == 0)
      return true;
   else
      return false;
}
/* возвращает true, если список полон */
bool ListIsFull(const List * plist)
{
   if (plist->items == MAXSIZE)
      return true;
   else
      return false;
}
/* возвращает количество узлов */
unsigned int ListItemCount(const List * plist)
{
   return plist->items;
}
/* добавляет элемент в список                                    */
/* предполагается, что для типа Item определена операция =  */
bool AddItem(Item item, List * plist)
{
   if (plist->items == MAXSIZE)
      return false;
   else
   {
      plist->entries[plist->items++] = item;
      return true;
   }
}
/* посещает каждый узел и выполняет функцию, указанную pfun */
void Traverse (const List * plist, void (* pfun)(Item item) )
{
   int i;
   for (i = 0; i < plist->items; i++)
      (*pfun)(plist->entries[i]); /* применить функцию к элементу списка */
}
/* функция malloc() не использовалась, поэтому освобождать нечего */
/* устанавливает член items в 0                                    */
void EmptyTheList(List * plist)
{
   plist->items = 0;
}
```

17.5.

```
/* pe17-5.h -- заголовочный файл для типа stack */
#ifndef STACK_H_
#define STACK_H_
#include <stdbool.h> /* C99 */
/* enum bool {false, true}; */ /* до выхода C99*/

/* ПОМЕСТИТЕ СЮДА ОПРЕДЕЛЕНИЕ ТИПА ЭЛЕМЕНТА    */
/* НАПРИМЕР, typedef int Item;                 */

typedef char Item;
#define MAXSTACK 100
typedef struct stack
{
   Item items[MAXSTACK]; /* содержит сведения о стеке         */
   int top;              /* индекс первой пустой ячейки       */
} Stack;

/* операция:    инициализация стека                           */
/* предусловие: ps указывает на стек                          */
/* постусловие: стек инициализирован пустым значением         */
void InitializeStack(Stack * ps);

/* операция:    проверяет, является ли стек полным            */
/* предусловие: ps указывает на ранее инициализированный стек */
/* постусловие: возвращает значение true, если стек полон,    */
/*              иначе возвращает значение false               */
bool FullStack(const Stack * ps);

/* операция:    проверяет, является ли стек пустым            */
/* предусловие: ps указывает на ранее инициализированный стек */
/* постусловие: возвращает значение true, если стек пуст,     */
/*              иначе возвращает значение false               */
bool EmptyStack(const Stack *ps);

/* операция:    заталкивает элемент в стек                    */
/* предусловие: ps указывает на ранее инициализированный стек */
/*              элемент должен помещаться                     */
/*              в верхушку стека                              */
/* постусловие: если стек не полон, элемент помещается        */
/*              в верхушку стека и функция возвращает         */
/*              значение true; иначе стек остается            */
/*              неизменным, а функция возвращает              */
/*              значение false                                */
bool Push(Item item, Stack * ps);

/* операция:    удаляет элемент из верхушки стека             */
/* предусловие: ps указывает на ранее инициализированный стек */
/* постусловие: если стек не пуст, элемент в верхушке         */
/*              стека копируется в *pitem и удаляется         */
/*              из стека, а функция возвращает                */
/*              значение true; если операция                  */
/*              опустошает стек, стек                         */
/*              переустанавливается в пустое состояние.       */
/*              Если стек пуст с самого начала, он            */
/*              остается неизменным, а функция                */
/*              возвращает значение false                     */
bool Pop(Item *pitem, Stack * ps);
#endif

/* pe17-5a.c */
#include <stdio.h>
```

```
#include <string.h>
#include "pe17-5.h"
#define SLEN 81
char * s_gets(char * st, int n);
int main(void)
{
   Stack stch;
   char temp[SLEN];
   int i;
   char ch;
   InitializeStack(&stch);
   printf("Введите строку (пустая строка приведет к завершению): \n");
   while (s_gets(temp, SLEN) && temp[0] != '\0')
   {
      i = 0;
      while (temp[i] != '\0' && !FullStack(&stch))
         Push(temp[i++], &stch);
      while (!EmptyStack(&stch))
      {
         Pop(&ch, &stch);
         putchar(ch);
      }
      putchar('\n');
      printf("Введите следующую строку (пустая строка приведет к завершению): ");
   }
   puts("Программа завершена.");
   return 0;
}

char * s_gets(char * st, int n)
{
   char * ret_val;
   char * find;
   ret_val = fgets(st, n, stdin);
   if (ret_val)
   {
      find = strchr(st, '\n');  // поиск новой строки
      if (find)                 // если адрес не равен NULL,
         *find = '\0';          // поместить туда нулевой символ
      else
         while (getchar() != '\n')
            continue;           // отбросить остаток строки
   }
   return ret_val;
}

/* pe17-5b.c -- операции над стеком */
#include <stdio.h>
#include <stdlib.h>
#include "pe17-5.h"

void InitializeStack(Stack * ps)
{
   ps->top = 0;
}

bool FullStack(const Stack * ps)
{
   return ps->top == MAXSTACK;
}
```

```
bool EmptyStack(const Stack *ps)
{
   return ps->top == 0;
}
bool Push(Item item, Stack * ps)
{
   if (ps->top == MAXSTACK)
      return false;
   else
   {
      ps->items[ps->top++] = item;
      return true;
   }
}
bool Pop(Item *pitem, Stack * ps)
{
   if (ps->top == 0)
      return false;
   else
   {
      ps->top--;
      *pitem = ps->items[ps->top];
      return true;
   }
}
```

17.6.

```
/* pe17-6.c */
#include <stdio.h>
int inarray(const int sorted[], int size, int val);
#define SIZE 10
int main(void)
{
   int nums[SIZE] = {1, 20, 40, 41, 42, 43, 70, 88, 92, 109};
   int num;
   int found;
   printf ("Введите целое число, которое нужно найти: ");
   while (scanf("%d", &num) == 1)
   {
      found = inarray(nums, SIZE, num);
      printf ("%d %s в массиве.\n", num, found? "нашлось" : "не нашлось");
      printf("Введите следующее целое число (или q для завершения): ");
   }
   printf("Программа завершена.\n");
   return 0;
}
int inarray(const int sorted[], int size, int val)
{
   int min = 0;
   int max = size - 1;
   int mid;
   int found = 0;
   while (min < max)
   {
      mid = (min + max) / 2;
      if (val < sorted[mid])
         max = mid - 1;
```

```
        else if (val > sorted[mid])
            min = mid + 1;
        else
        {
            found = 1;
            break;
        }
    }
    if (sorted[min] == val)
        found = 1;
    return found;
}
```

17.7.

```
/* pe17-7.h: копия tree.h -- двоичное дерево поиска  */
/* в этом дереве дублированные элементы не разрешены */
#ifndef _TREE_H_
#define _TREE_H_
#include <stdbool.h> /* C99 */
/* enum bool {false, true}; */ /* до выхода C99 */
#define SLEN 81
/* переопределите Item подходящим образом */
typedef struct item
{
    char wrd[SLEN];
    int count;
} Item;
#define MAXITEMS 100
typedef struct node
{
    Item item;
    struct node * left;  /* указатель на правую ветвь */
    struct node * right; /* указатель на правую ветвь */
} Node;
typedef struct tree
{
    Node * root;       /* указатель на корень дерева    */
    int size;          /* количество элементов в дереве */
} Tree;

/* прототипы функций */

/* операция:     инициализация дерева пустым содержимым     */
/* предусловия:  ptree указывает на дерево                  */
/* постусловия:  дерево установлено в пустое состояние      */
void InitializeTree(Tree * ptree);

/* операция:     определение, является ли дерево пустым     */
/* предусловия:  ptree указывает на дерево                  */
/* постусловия:  функция возвращает true, если дерево       */
/*               пустое, и false - в противном случае       */
bool TreeIsEmpty(const Tree * ptree);

/* операция:     определение, является ли дерево полным     */
/* предусловия:  ptree указывает на дерево                  */
/* постусловия:  функция возвращает true, если дерево       */
/*               полное, и false - в противном случае       */
bool TreeIsFull(const Tree * ptree);

/* операция:     определение количества элементов в дереве  */
/* предусловия:  ptree указывает на дерево                  */
```

```
/* постусловия:  функция возвращает количество элементов в дереве       */
int TreeItemCount(const Tree * ptree);

/* операция:     добавление элемента к дереву                            */
/* предусловия:  pi – адрес добавляемого элемента                        */
/*               ptree указывает на инициализированное дерево            */
/* постусловия:  если возможно, функция добавляет элемент                */
/*               к дереву и возвращает true;                             */
/*               в противном случае она возвращает false                 */
bool AddItem(const Item * pi, Tree * ptree);

/* операция:     поиск элемента в дереве                                 */
/* предусловия:  pi указывает на элемент                                 */
/*               ptree указывает на инициализированное дерево            */
/* постусловия:  функция возвращает true, если элемент присутствует      */
/*               в дереве, и false – в противном случае                  */
bool InTree(const Item * pi, const Tree * ptree);

/* операция:     удаление элемента из дерева                             */
/* предусловия:  pi – адрес удаляемого элемента                          */
/*               ptree указывает на инициализированное дерево            */
/* постусловия:  если возможно, функция удаляет элемент из дерева        */
/*               и возвращает true; в противном случае функция           */
/*               возвращает false                                        */
bool DeleteItem(const Item * pi, Tree * ptree);

/* операция:     применение указанной функции к каждому элементу в дереве*/
/* предусловия:  ptree указывает на дерево                               */
/*               pfun указывает на функцию, которая принимает            */
/*               аргумент Item и не имеет возвращаемого значения         */
/* постусловия:  функция, указанная с помощью pfun, выполняется один раз */
/*               для каждого элемента в дереве                           */
void Traverse(const Tree * ptree, void (* pfun)(Item item));

/* операция:     удаление всех элементов из дерева                       */
/* предусловия:  ptree указывает на инициализированное дерево            */
/* постусловия:  дерево является пустым                                  */
void DeleteAll(Tree * ptree);

/* операция:    возвращение адреса элемента в дереве                     */
/* предусловия: pi указывает на элемент                                  */
/*              ptree указывает на инициализированное дерево             */
/* постусловия: функция возвращает адрес, если элемент находится        */
/*              в дереве, и NULL в противном случае                      */
const Item * WhereInTree(const Item * pi, const Tree * ptree);
#endif
```

```
/* pe17-7a.c */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "pe17-7.h"
#define SLEN 81
void printitem(Item item);
char menu(void);
void showwords (const Tree * pt);
void findword (const Tree * pt);
char * s_gets(char * st, int n);
int main(void)
{
```

```
    Tree wordcount;
    FILE * fp;
    char filename[SLEN];
    char word[SLEN];
    Item entry;
    char choice;
    printf("Введите имя файла, подлежащего обработке: \n");
    s_gets(filename, SLEN);
    if ((fp = fopen(filename, "r")) == 0)
    {
       printf("Не удается открыть файл %s. Программа завершена.\n", filename);
       exit(EXIT_FAILURE);
    }
    InitializeTree(&wordcount);
    while (fscanf(fp, "%s", word) == 1 && !TreeIsFull(&wordcount))
    {
       strcpy(entry.wrd, word);
       AddItem(&entry, &wordcount);
    }
    while ((choice = menu()) != 'q')
    {
       switch (choice)
       {
          case 's' : showwords(&wordcount);
                     break;
          case 'f' : findword(&wordcount);
                     break;
          default : puts("ошибка в switch");
       }
    }
    fclose(fp);
    puts("Программа завершена.");
    return 0;
}

char * s_gets(char * st, int n)
{
    char * ret_val;
    char * find;
    ret_val = fgets(st, n, stdin);
    if (ret_val)
    {
       find = strchr(st, '\n'); // поиск новой строки
       if (find)                // если адрес не равен NULL,
          *find = '\0';         // поместить туда нулевой символ
       else
          while (getchar() != '\n')
             continue;          // отбросить остаток строки
    }
    return ret_val;
}

char menu(void)
{
    int ch;
    puts("Программа подсчета слов");
    puts("Введите букву, которая соответствует интересующему варианту:");
    puts("s) показать список слов  f) найти слово");
    puts("q) выйти из программы");
```

```
    while ((ch = getchar()) != EOF)
    {
        while (getchar() != '\n') /* отбросить оставшуюся часть строки */
            continue;
        ch = tolower(ch);
        if (strchr("sfq",ch) == NULL)
            puts("Введите s, f или q:");
        else
            break;
    }
    if (ch == EOF) /* обнаружение EOF приводит к завершению программы */
        ch = 'q';
    return ch;
}

void showwords (const Tree * pt)
{
    if (TreeIsEmpty(pt))
        puts("Записи отсутствуют.");
    else
        Traverse(pt, printitem);
}

void findword (const Tree * pt)
{
    char word[SLEN];
    Item entry;
    const Item * pi;
    if (TreeIsEmpty(pt))
    {
        puts("Записи отсутствуют.");
        return; /* выйти из функции, если дерево пустое */
    }
    printf("Введите слово для поиска: ");
    scanf("%s", word);
    while (getchar() != '\n')
        continue;
    strcpy(entry.wrd, word);
    pi = WhereInTree(&entry, pt);
    if (pi == NULL)
        printf("%s не находится в списке.\n", word);
    else
        printf("%s встречается %d раз(а).\n", word, pi->count);
}

void printitem(Item item)
{
    printf("%3d: %s\n", item.count,
           item.wrd);
}

/* pe17-7b.c: копия tree.c -- функции поддержки дерева */
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "pe17-7.h"

/* локальный тип данных */
typedef struct pair {
    Node * parent;
    Node * child;
} Pair;
```

```
/* прототипы для локальных функций */
static Node * MakeNode(const Item * pi);
static bool ToLeft(const Item * i1, const Item * i2);
static bool ToRight(const Item * i1, const Item * i2);
static void AddNode (Node * new_node, Node * root);
static void InOrder(const Node * root, void (* pfun)(Item item));
static Pair SeekItem(const Item * pi, const Tree * ptree);
static void DeleteNode(Node **ptr);
static void DeleteAllNodes(Node * ptr);

/* определения функций */
void InitializeTree(Tree * ptree)
{
   ptree->root = NULL;
   ptree->size = 0;
}

bool TreeIsEmpty(const Tree * ptree)
{
   if (ptree->root == NULL)
      return true;
   else
      return false;
}

bool TreeIsFull(const Tree * ptree)
{
   if (ptree->size == MAXITEMS)
      return true;
   else
      return false;
}

int TreeItemCount(const Tree * ptree)
{
   return ptree->size;
}

bool AddItem(const Item * pi, Tree * ptree)
{
   Node * new;
   Pair seek;
   if (TreeIsFull(ptree))
   {
      fprintf(stderr,"Дерево заполнено.\n");
      return false;      /* преждевременный возврат     */
   }
   if ((seek = SeekItem(pi, ptree)).child != NULL)
   {
      seek.child->item.count++;
      return true;       /* преждевременный возврат     */
   }
   new = MakeNode(pi);   /* new указывает на новый узел  */
   if (new == NULL)
   {
      fprintf(stderr, "Не удалось создать узел.\n");
      return false;      /* преждевременный возврат */
   }
   /* успешное создание нового узла */
   ptree->size++;
```

```
    if (ptree->root == NULL)        /* случай 1: дерево пустое      */
       ptree->root = new;           /* новый узел - корень дерева   */
    else                            /* случай 2: дерево не пустое   */
       AddNode(new,ptree->root);    /* добавление узла к дереву     */
    return true;
}

bool InTree(const Item * pi, const Tree * ptree)
{
    return (SeekItem(pi, ptree).child == NULL) ? false : true;
}

const Item * WhereInTree(const Item * pi, const Tree * ptree)
{
    Node * pn;
    pn = SeekItem(pi,ptree).child;
    if (pn != NULL)
       return &(pn->item);
    else return NULL;
}

bool DeleteItem(const Item * pi, Tree * ptree)
{
    Pair look;
    look = SeekItem(pi, ptree);
    if (look.child == NULL)
       return false;
    if (look.child->item.count > 0)
       look.child->item.count--;
    else
    {
       if (look.parent == NULL)     /* удаление корневого элемента  */
          DeleteNode(&ptree->root);
       else if (look.parent->left == look.child)
          DeleteNode(&look.parent->left);
       else
          DeleteNode(&look.parent->right);
       ptree->size--;
    }
    return true;
}

void Traverse (const Tree * ptree, void (* pfun)(Item item))
{
    if (ptree != NULL)
    InOrder(ptree->root, pfun);
}

void DeleteAll(Tree * ptree)
{
    if (ptree != NULL)
       DeleteAllNodes(ptree->root);
    ptree->root = NULL;
    ptree->size = 0;
}

/* локальные функции */
static void InOrder(const Node * root, void (* pfun)(Item item))
{
    if (root != NULL)
    {
       InOrder(root->left, pfun);
```

```
        (*pfun)(root->item);
        InOrder(root->right, pfun);
    }
}
static void DeleteAllNodes(Node * root)
{
    Node * pright;
    if (root != NULL)
    {
        pright = root->right;
        DeleteAllNodes(root->left);
        free(root);
        DeleteAllNodes(pright);
    }
}
static void AddNode(Node * new_node, Node * root)
{
    if (ToLeft(&new_node->item, &root->item))
    {
        if (root->left == NULL)    /* пустое поддерево,          */
            root->left = new_node; /* поэтому добавить сюда узел */
        else
            AddNode(new_node, root->left); /* иначе обработать поддерево */
    }
    else if (ToRight(&new_node->item, &root->item))
    {
        if (root->right == NULL)
            root->right = new_node;
        else
            AddNode(new_node, root->right);
    }
    else /* дубликаты не разрешены */
    {
        fprintf(stderr, " Ошибка местоположения в AddNode().\n");
        exit(1);
    }
}
static bool ToLeft(const Item * i1, const Item * i2)
{
    if (strcmp(i1->wrd, i2->wrd) < 0)
        return true;
    else
        return false;
}
static bool ToRight(const Item * i1, const Item * i2)
{
    if (strcmp(i1->wrd, i2->wrd) > 0)
        return true;
    else
        return false;
}
static Node * MakeNode(const Item * pi)
{
    Node * new_node;
    new_node = (Node *) malloc(sizeof(Node));
    if (new_node != NULL)
    {
```

```
        new_node->item = *pi;
        new_node->item.count = 1;
        new_node->left = NULL;
        new_node->right = NULL;
    }
    return new_node;
}
static Pair SeekItem(const Item * pi, const Tree * ptree)
{
    Pair look;
    look.parent = NULL;
    look.child = ptree->root;
    if (look.child == NULL)
        return look; /* преждевременный возврат */
    while (look.child != NULL)
    {
        if (ToLeft(pi, &(look.child->item)))
        {
            look.parent = look.child;
            look.child = look.child->left;
        }
        else if (ToRight(pi, &(look.child->item)))
        {
            look.parent = look.child;
            look.child = look.child->right;
        }
        else /* если элемент не расположен ни слева, ни справа, он должен быть таким же */
            break;   /* look.child - это адрес узла, содержащего элемент */
    }
    return look;   /* возврат в случае успеха */
}
static void DeleteNode(Node **ptr)
/* ptr - это адрес родительского элемента, указывающего на целевой узел */
{
    Node * temp;
    if ((*ptr)->left == NULL)
    {
        temp = *ptr;
        *ptr = (*ptr)->right;
        free(temp);
    }
    else if ( (*ptr)->right == NULL)
    {
        temp = *ptr;
        *ptr = (*ptr)->left;
        free(temp);
    }
    else /* удаляемый узел имеет два дочерних узла */
    {
        /* выяснение места присоединения правого поддерева */
        for (temp = (*ptr)->left; temp->right != NULL;
             temp = temp->right)
            continue;
        temp->right = (*ptr)->right;
        temp = *ptr;
        *ptr =(*ptr)->left;
        free(temp);
    }
}
```