А.А. Мелещенко

Основы программирования на языке С. Практикум

Версия 3.0 (10.02.10).

Содержание

Практические аспекты разработки программ

"Единственный способ изучать новый язык программирования – писать на нем программы".

Брайэн Керниган

Обычная ошибка начинающих программистов состоит в том, что, получив задание, они тут же садятся за компьютер и начинают быстро набирать код. Переменным даются первые попавшиеся имена, когда программа вдруг не работает так, «как нужно», ее структура переделывается заново, иногда по нескольку раз. После того, как объем кода превысит некую критическую отметку, программа начинает работать нестабильно и сам автор порою не понимает, что в ней происходит.

Работает такая программа обычно только «в руках автора» на нескольких подобранных им наборах данных. Внесение даже небольших изменений (как и ввод «не авторского» набора данных) приводит к сбоям в работе. В этих случаях можно услышать, что «дома все работало», «виноват компьютер» или «операционная система», часто упоминается Билл Гейтс.

В компаниях, разрабатывающих программное обеспечение, за такой стиль программирования «бьют по рукам». А с теми, кто уж сильно пристрастился к нему, приходится расставаться.

Для того чтобы ваш подход к разработке программного обеспечения с самого начала был профессиональным, обратите внимание на следующие ключевые моменты:

1. Проектирование.
2. Тестирование.
3. Метод пошаговой разработки.
4. Хороший стиль написания программ.
5. Оценка сроков разработки.

Проектирование

Готовясь к написанию программы, уделите достаточное время *проектированию*. В случае небольших задач для этого достаточно карандаша и одного чистого листа бумаги. Не садитесь за компьютер, пока решение задачи не будет для вас ясным, а все основные случаи и «ветви» алгоритма – продуманными и формализованными.

Структура программы должна быть стройной и логичной. Избегайте двусмысленности и повторения функциональных частей в вашем проекте, стремитесь к простоте. Если есть несколько способов решения задачи, следует выбрать самый простой из них. Не забывайте, что ошибки на стадии проектирования самые «дорогие», на их исправление тратится больше всего времени. Не спешите. Исследование компании IBM показало, что каждый час, затраченный на анализ качества и проектирование, экономит от 3 до 10 часов работы.

Стив Макконнелл (Steve McConnell), специалист компании Microsoft, приводит следующую оценку трудоемкости этапов разработки ПО:

Таблица 1

Этап работы	Малый проект (2,500 строк)*	Большой проект (500,000 строк)*
Проектирование архитектуры (описание системы с точки зрения пользователя)	10%	30%
Подробный проект (спецификации)	20%	20%
Кодирование и отладка	25%	10%
Тестирование модулей	20%	5%
Интеграция (объединение модулей в систему)	15%	20%

* Данные взяты на основе статистических данных 4000 программных проектов, выполнявшихся в США и Европе.

Обратите внимание, что проектирование занимает 30% времени в малом проекте и 50% в большом!

Тестирование

Если программа работает, и делает то, что задумано программистом, задача выполнена лишь наполовину. Хорошо протестированная программа должна работать при любом наборе корректных входных данных, а во всех остальных случаях – выводить пользователю соответствующие сообщения.

Синтаксические ошибки с разной степенью эффективности выявляются трансляторами. Ваша задача – убедиться в корректности алгоритма, то есть получить уверенность, что программа выдает правильные результаты на всех «ветвях» алгоритма.

Доказано, что ни теоретически, ни посредством тестирования невозможно доказать правильность программы – можно лишь спровоцировать появление и устранить в ней как можно больше ошибок. Причем процесс обнаружения ошибок подчиняется закону насыщения, то есть большинство ошибок обнаруживается на ранних стадиях тестирования,

и чем меньше в программе осталось ошибок, тем дольше нужно искать каждую из них.

Тест – совокупность исходных данных для программы вместе с ожидаемыми результатами. В наборе тестов выделяют три группы:

- «тепличные» – проверяющие программу при корректных, нормальных исходных данных самого простого вида;
- «экстремальные» – на границе области определения в ситуациях, которые могут произойти и на которые нужно корректно реагировать;
- «запредельные» – за границей области определения (а, возможно, и здравого смысла) – ситуации, бессмысленные с точки зрения постановки задачи, но которые могут произойти из-за ошибок пользователя или внешних программ.

Хорошие программы должны быть протестированы на всех трех группах тестов. Вопрос «на засыпку»: что делать, если на каком-то тесте из вашего набора выявилась ошибка? Правильно, весь набор тестов нужно «прогонять» заново.

Тестирование должно осуществляться *параллельно* с написанием программного кода. Пренебрегая тестированием на ранних стадиях, вам будет сложно добиться правильной работы программы позже, так как количество комбинаций, которые могут вызвать ошибку при соединении непротестированных компонентов, растет экспоненциально.

Пошаговая разработка программ

Бесчисленное количество программистов на своем опыте убедилось, что крупные проекты не рождаются сразу – они вырастают из небольших систем. Как говорят специалисты, программы нужно не *строить*, и уж тем более не *писать*, а *выращивать* методом пошаговой разработки.

По мнению ряда экспертов (см., например, статью Ф.Брукса «Серебряной пули нет») метод пошаговой разработки является достаточно простым и, в то же время, способен значительно повысить производительность труда программиста. Пошаговая разработка приносит ощутимые выгоды как в больших проектах, так и в малых, поэтому этот метод вполне подходит и для выполнения лабораторных работ.

Суть метода. Вначале нужно просто заставить программу выполняться, пусть даже она не делает ничего полезного. Другими словами, заставьте компьютер напечатать «Hello» на экране (иногда и это не простая задача, т.к. нужно «повозиться» с настройками, подключением библиотек и пр.). Затем объявите несколько основных функций и заставьте их выполняться в нужной последовательности, формируя, таким образом, каркас программы. Функции пока не должны иметь «тела», т.е. должны представлять собой «заглушки». Если все хорошо и программа работает,

постепенно начинайте наполнять кодом каждую функцию, сопровождая этот процесс тестированием.

Точно также поступайте и в отношении функций более низкого уровня, которые вызываются из ранее реализованных. Вначале просто опишите их и заставьте выполняться, а потом постепенно наращивайте функциональность, не забывая о тестировании.

Метод пошаговой разработки делает процесс разработки более стабильным и предсказуемым. Каждый день вы будете иметь рабочую версию программы. Реализуйте вначале самые главные функции, затем – менее важные. «Выращивайте» вашу программу каждый день, пока она не будет полностью удовлетворять поставленным требованиям.

В начале такой подход может показаться слишком медленным, но на практике он оказывается самым быстрым. Плохая практика – сразу написать много кода, а потом пытаться заставить программу работать «как следует». В этом случае даже в коротких модулях на отладку может уйти масса времени. В малых и крупных программных проектах такой подход ведет к срыву проекта и (очень часто) переписыванию всей системы заново.

Хороший стиль программирования

Вряд ли вы всю жизнь будете программировать в одиночку. Вы будете работать в коллективе программистов. Хорошо, если с самого начала вы привыкните писать программы в общепринятом стиле, что облегчит их понимание другими людьми (и вами самими).

Ниже приводится пример кода, выдержанного в хорошем стиле:

```
#include <stdio.h>
#define MAX 16000

int Sum(int n)
{
  int i, total = 0;

  for (i = 0; i < n; i++) {
    total += i;
    if (total > MAX) {
      printf("Too large!\n");
      exit(1);
    }
  }
  return total;
}
```

Вот некоторые «негласные» требования хорошего стиля:

1. Один оператор на строку.
2. Пустая строка после начальных объявлений.
3. После знаков препинания, до и после каждого оператора используются пробелы.
4. Открывающая фигурная скобка составного оператора ставится в той же строке, что и оператор. Завершающая фигурная скобка выровнена по линии начального символа оператора.
5. Все, что встречается после открытия открывающей фигурной скобки, выравнивается на заданное число пробелов; например, в этом примере используется два пробела. Соответствующая закрывающая фигурная скобка ставится так, что последующие операторы выравниваются по ней.
6. Идентификаторы препроцессора и константы печатаются заглавными буквами. Обычные идентификаторы – строчными.
7. Команды препроцессора и объявления на уровне файла записываются без отступов слева.

Часто встречающиеся разновидности этого стиля включают:

1. Выравнивание фигурных скобок всегда с новой строки.
2. Короткие операторы, которые концептуально связаны, могут находиться в одной и той же строке.

```
/* Модификации стиля */
int Fibonacci(int n)
{
  int x = 1, y = 1, z = 1, i = 2;

  while (i < n)
  {
    z = x + y; x = y; y = z;
    i++;
  }
  return z;
}
```

Оценка сроков разработки

Вернемся еще раз к табл. 1. Собранные в ней данные реальных проектов говорят и том, почему начинающие программисты так часто ошибаются в оценке сроков разработки программ. Они, как правило, полагают, что основным из этапов разработки является *кодирование*, а продолжительность всего проекта примерно равна времени, нужному на написание и отладку кода. В результате, они сильно занижают сроки разработки (иногда в несколько раз), поскольку, как видно из статистических

данных, время на кодирование составляет в небольших проектах лишь одну четвертую, а в больших – одну десятую от всего времени проекта.

Вторая ошибка заключается в том, что часто планируют на случай, что «все будет хорошо»: в программе не появятся трудноуловимые ошибки, поиск и устранение которых займет массу времени, не будет проблем с системным программным обеспечением, с компьютером, с интернетом, никто не заболеет, не появятся срочные отвлекающие дела и т.д. Каждое из этих событий, конечно, может и не случиться, но вероятность того, что не произойдет ни одно из них, в случае достаточно объемной разработки стремится к нулю. В реальных программных проектах процент времени, уходящего на «незапланированные мероприятия», достигает огромных величин и в среднем равен 30%.

Таким образом, для небольшого проекта (2-4 тыс. строк) можно использовать следующий несложный метод оценки:

1. На основе своего опыта оцените время, необходимое, чтобы запрограммировать данную задачу.
2. Умножьте полученную величину на четыре (если программа небольшая, менее 2 тыс. срок, то на два).
3. Добавьте 30% времени на непредвиденные случаи.

Полученная оценка и будет реальным сроком, хотя и она может оказаться слишком оптимистичной.

Умение правильно оценить сроки разработки программного обеспечения приходит только с многолетним опытом и является отличительной чертой профессиональных программистов.

1. Основные конструкции

типы данных, арифметические операторы, циклы.

Контрольные вопросы

1. С какой функции начинается каждая программа на C?
2. В чем сходство и различие директив **#include** и **#define**? Чем различаются записи **#include <my.h>** и **#include "my.h"**?
3. Какие символы могут включать идентификаторы в C? Какой длины они могут быть? Являются ли имена **count** и **Count** разными идентификаторами?
4. Какие целые типы данных вы знаете? Укажите диапазон значений каждого типа данных. Что такое «эффект сворачивания» целой переменной? В чем разница между переменными **int i;** и **unsigned int j;**? В каких случаях предпочтительнее использовать переменную **j**?
5. Какие типы данных с плавающей запятой вы знаете? Какой диапазон значений они имеют? Почему переменные с плавающей запятой не могут представить математические числа с абсолютной точностью?
6. Объясните, как хранятся символы в памяти компьютера. Что представляет собой строка в C?
7. Объясните различие между переменной, константой и символической константой. В каких случаях их целесообразно использовать?
8. В каких случаях используют перечисление (**enum**)? Каким образом компилятор нумерует элементы в объявлении **enum**? Объясните, как операторы **enum** и **typedef** помогают сделать код программы более понятным, приведите примеры.
9. Каковы плюсы и минусы использования выражений со смешанными типами данных? В каких случаях из-за преобразования типов может быть потеряна информация? Что в этом случае предпримет компилятор?
10. Какие арифметические, логические операторы и операторы отношений в языке C вы знаете? Объясните назначение оператора отрицания. Какое целое число в языке C соответствует значению «истина»? Значению «ложь»? Каким образом программист может указать нужный ему порядок вычислений?
11. Какие сокращенные операторы присваивания вы знаете? Объясните, как работают операторы инкремента и декремента. В чем заключается различие между постфиксной (оператор стоит после переменной) и префиксной (оператор стоит перед переменной) записи?
12. Чем похожи и чем различаются операторы **while**, **do-while** и **for**? В каких случаях предпочтительнее использовать каждый из них? Почему для

организации цикла не рекомендуется использовать оператор **goto**? Чем похожи и чем различаются операторы **break** и **continue**?

13. Переменные **x** и **y** имеют тип **int**. Какое значение они имеют в каждом случае?
    1) `x = (12 + 6) / 2 * 3;`
    2) `x = (int) 3.8 + 3.3;`
    3) `x = 3 / 5 * 22.0;`
    4) `x = 22.0 * 3 / 5;`
    5) `y = x = (2 + 3) / 4;`
    6) `y = 3 + 2 * (x = 7 / 2);`
14. Какие из следующих выражений истинны, а какие ложны?
    1) `3 + 4 > 2 && 3 < 2;`
    2) `100 > 3 && 'a' > 'c';`
    3) `100 > 3 || 'a' > 'c';`
    4) `!(100 > 3);`
    5) `x >= y || y > x;`
    6) `'X' > 'T' ? 10 : 5;`
    7) `x > y ? y > x : x > y;`
15. Какие ошибки содержит эта программа?

```
main() / это идеальная программа /
{
  integer i = 1;
  double n;
  printf("Далее — последовательность дробей\n);
  while (i < 30)
    n = 1 / i;
    printf(" %lf", n);
  printf("Конец работы\n");
}
```

Примеры решения задач

Пример 1.1. Определить первую и последнюю цифры натурального числа *n*.

ex11.c

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

void main(void)
{
  long n, first, last;

  do {
    printf("\nВведите натуральное число: ");
    scanf("%ld", &n);
    first = n;
    last = n % 10;    /* вычисляется остаток от деления */
```

```
                    /* введенного числа на 10. Он и будет */
                    /* являться последней цифрой. */
    while (first > 9) /* последовательно делим число на 10 */
       first /= 10;   /* нацело, пока в first не сохранится */
                    /* первая цифра числа */
    printf("\nВ числе %ld первая цифра - %ld, последняя - %ld",
      n, first, last);
    printf("\nПродолжим? (y/n): ");
  } while (getch() == 'y');
}
```

Пример 1.2. Даны вещественное число x и натуральное n. Вычислить

$$\sum_{k=1}^{n}(-1)^k \frac{x^k}{(2k+1)!}$$

Для вычисления указанной суммы приведены три фрагмента. Первый демонстрирует неэффективный подход, когда задача решается «в лоб»:

```
...
int n, i, k;
double fact, x, s = 0.0;

printf("Введите количество слагаемых: ");
scanf("%d", &n);
printf("Введите вещественное число x: ");
scanf("%lf", &x);
for (k = 1; k <= n; k++) {
  for (i = 1, fact = 1; i <= 2*k + 1; i++)
    fact *= i;
  s += pow(-1.0, (double)k) * pow(x, (double)k) / fact;
}
```

Недостаток этой программы состоит в том, что в ней не учитываются рекуррентные зависимости, которые можно использовать при вычислении факториала и степени. Кроме того, совсем не обязательно для вычисления $(-1)^k$ использовать громоздкую функцию **pow(),** т.к. это отнимает время.

Программу можно существенно улучшить, если воспользоваться такими соотношениями:

$$(-1)^k = \begin{cases} 1, & при\ k, кратном\ 2; \\ -1, при\ k, не\ кратном\ 2; \end{cases}$$

$$x^k = x^{k-1} \times x;$$

$$(2k+1)! = (2(k-1)+1)! * 2k * (2k+1)$$

вычислено на предыдущем шаге

С учетом этого запишем второй вариант программы:

```
...
int n, k, ed = 1;
double x, s = 0.0, fact = 1.0, stepX = 1.0;

printf("Введите количество слагаемых: ");
scanf("%d", &n);
printf("Введите вещественное число x: ");
scanf("%lf", &x);
for (k = 1; k <= n; k++) {
  ed = -ed;
  stepX *= x;
  fact *= 4*k*k + 2*k;
  s += ed * stepX / fact;
}
```

Недостатком обоих приведенных фрагментов является то, что в них степень и факториал числа считаются по отдельности. Как известно, функции факториала и степени растут достаточно быстро, поэтому уже при довольно небольших значениях *n* в программе произойдет переполнение и она будет работать неправильно. В данном случае лучше на очередном шаге цикла во время вычисления степени и факториала сразу делить их друг на друга, чтобы они взаимно «гасились». С учетом этого запишем последний вариант программы:

ex12.c

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

void main(void)
{
  int n, k, ed = 1;
  double x, slag = 1, s = 0;

  printf("Введите количество слагаемых: ");
  scanf("%d", &n);
  printf("Введите вещественное число  x: ");
  scanf("%lf", &x);
  for (k = 1; k <= n; k++) {
    ed = -ed;
    slag *= x / (2*k * (2*k + 1));
    s += ed * slag;
  }
  printf("\ns = %lg", s);
}
```

Задачи

Требования к программам

Хорошо написанные программы отвечают следующим требованиям:

1. Работа программы должна точно удовлетворять условиям задачи (или, как еще говорят, техническому заданию). Следуйте простому правилу, которое поможет вам и в профессиональной карьере: делайте именно то, что требуется: не больше и не меньше. И чем проще – тем лучше.
2. Программа не должна содержать предупреждений (Warnings) компилятора.
3. Программа должна быть хорошо протестирована: при корректных наборах входных данных она должна работать правильно, в других случаях – выдавать соответствующие предупреждения.
4. Программа должна иметь дружественный интерфейс. Это значит, что пользователь, знакомый с условием задачи, должен «справиться» с программой самостоятельно, без дополнительных инструкций со стороны программиста.
5. Код программы должен быть выдержан в хорошем стиле программирования, идентификаторы – иметь осмысленные имена. Желательны комментарии в ключевых местах программы.

Задача 1.1

1.1 (1). **Шахматы.** Даны натуральные числа k, l, m, n. Выяснить, угрожает ли конь, стоящий на поле (k, l) шахматной доски, полю (m, n).

1.1 (2). **Погашение кредита.** Предприниматель, начав дело, взял кредит размером k рублей под p процентов годовых и вложил его в свое дело. По прогнозам, его дело должно давать прибыль r рублей в год. Сможет ли он накопить достаточную сумму, достаточную для погашения кредита, и если да, то через сколько лет?

1.1 (3). **Расписание звонков.** В школе задается начало учебного дня, продолжительность перерывов и количество уроков. Продолжительность урока составляет 45 минут. Получить расписание звонков на весь учебный день.

1.1 (4). **Селекция.** Селекционер вывел новый сорт зерновой культуры и снял с опытной делянки k кг семян. Посеяв 1 кг семян, можно за сезон собрать p кг семян. Через сколько лет селекционер сможет засеять новой культурой поле площадью s га, если норма высева n кг/га?

1.1 (5). **Квадраты**. Стороны прямоугольника заданы натуральными числами n и m. Найти количество квадратов, стороны которых выражены натуральными числами, на которые можно разрезать

данный прямоугольник, если от него каждый раз отрезается квадрат максимальной площади.

1.1 (6). **Часы и минуты.** С момента полночи прошло n минут. Определить угол в градусах между положением часовой и минутной стрелок часов в указанный момент времени.

1.1 (7). **Старинная задача**. Сколько можно купить быков, коров и телят, при условии, что плата за быка 10 рублей, за корову – 5 рублей, за теленка – полтинник (0,5 рубля). На 100 рублей надо купить 100 голов скота.

1.1 (8). **Вид треугольника.** Заданы три числа: a, b, c. Выяснить, можно ли построить треугольник с такими длинами сторон, и если можно, то какой это треугольник: равносторонний, равнобедренный, прямоугольный или общего вида.

1.1 (9). **Цена на молоко.** Фермер в начале каждой зимы повышает отпускную цену на молоко на $p\%$, а каждым летом – снижает на столько же процентов. Изменится ли цена на молоко и если да, то в какую сторону и на сколько через n лет?

1.1 (10). **Переправа.** Чапаеву надо под прямым углом к фарватеру преодолеть реку Урал шириной b м. Его скорость в стоячей воде v_1 м/с; скорость течения реки – v_2 м/с. Под каким углом к фарватеру он должен плыть, чтобы его «не снесло»? Сколько времени займет переправа? Как изменится решение, если посредине реки Чапаева ранили в руку, и его скорость с v_1 м/с упала до v_3 м/с?

1.1 (11). **Длина высоты.** Треугольник ABC задан длинами своих сторон. Найти длину высоты, опущенной из вершины A.
Экстремальные тесты: сумма двух сторон равна третьей; одна из сторон равна нулю.

1.1 (12). **Часы и минуты.** С начала суток часовая стрелка повернулась на y градусов (y – целое число от 0 до 360). Определить число целых часов и число полных минут, пошедших с начала суток.

1.1 (13). **Площадь.** Дано натуральное число q, выражающее площадь. Найти длины сторон (натуральные числа) всех таких прямоугольников (если они есть), площадь которых равна q.

1.1 (14). **Теорема Гольдбаха.** Математик Х. Гольдбах сформулировал гипотезу, согласно которой любое четное число, большее 2,

представимо в виде суммы 2 простых чисел. Проверить утверждение Гольдбаха для чисел, не превышающих 99.

1.1 (15). **Временной интервал.** Заданы моменты начала и конца некоторого промежутка времени в часах, минутах и секундах (в пределах одних суток). Найти продолжительность этого промежутка в тех же единицах измерения.

1.1 (16). **Коммерция.** Коммерсант, имея стартовый капитал k рублей, занялся торговлей, которая ежемесячно увеличивает капитал на $p\%$. Через сколько лет он накопит сумму s, достаточную для покупки собственного магазина?

1.1 (17). **Русские неметрические единицы длины:** 1 верста = 500 саженей; 1 сажень = 3 аршина; 1 аршин = 16 вершков; 1 вершок = 44,45 мм. Длина некоторого отрезка составляет p метров. Перевести ее в русскую неметрическую систему.

1.1 (18). **Шахматы.** Даны натуральные числа k, l, m, n. Требуется выяснить, угрожает ли ферзь, стоящий на поле (k, l) шахматной доски, полю (m, n).

1.1 (19). **Факультеты.** В БГУИР принято, что первая цифра номера студенческой группы означает последнюю цифру года поступления, следующая – номер факультета, следующие две – номер специальности, две последние цифры – номер группы на курсе. Продолжительность обучения – 5 лет. Дан номер группы студента БГУИР и текущий год. Напечатать, в каком году он поступил и на каком факультете учится.

Пример: гр. 252001, 2003 г. – факультет компьютерных систем и сетей, год поступления – 2002.

Справка. Номера факультетов: ФКП – 1, ФИТУ – 2, ФРЭ – 4, ФКСиС – 5, ФТК – 6, ЭФ – 7.

1.1 (20). **Из миль в километры.** Получить таблицу пересчета миль в километры и обратно (1 миля = 1,609344 км) для расстояний, не превышающих k км, в следующем виде:

мили	**километры**
0,6214	1,0000
1,0000	1,6093
1,2428	2,0000
1,8641	3,0000

1.1 (21). Мой возраст. Для заданного n: $0 \le n \le 200$, рассматриваемого как возраст человека, вывести фразу вида: «Мне 21 год», «Мне 32 года», «Мне 12 лет».

1.1 (22). Шахматы. Даны натуральные числа k, l, m, n. Требуется выяснить, являются ли поля (k, l) и (m, n) шахматной доски полями одного цвета.

1.1 (23). Расписание. Известно время начала и окончания (например, 6:00 и 24:00) работы некоторого пригородного автобусного маршрута с одним автобусом на линии, а также протяженность маршрута в минутах (в один конец) и время отдыха на конечных остановках. Составить суточное расписание этого маршрута (моменты отправления с конечных пунктов) без времени на обед и пересменку.

1.1 (24). Вырубка леса. Леспромхоз ведет заготовку древесины. Первоначальный объем ее на территории леспромхоза составлял p кубометров. Ежегодный прирост составляет $k\%$ Годовой план заготовки – t кубометров. Через сколько лет в бывшем лесу будут расти одни опята?

1.1 (25). Гуси и кролики. У гусей и кроликов вместе $2n$ лап. Сколько может быть гусей и кроликов? (Вывести все возможные сочетания).

1.1 (26). Обмен. Поменять местами значения целых переменных a и b, не используя дополнительные переменные.

1.1 (27). Часы и минуты. Даны целые числа h и m, указывающие момент времени: «h часов, m минут». Определить угол в градусах между положением часовой стрелки в начале суток и в указанный момент времени.

1.1 (28). Номер нарушителя. Три друга были свидетелями дорожно-транспортного происшествия. Первый заметил, что номер нарушителя делится на 2, 7 и 11. Второй запомнил, что в записи номера участвуют всего две различные цифры, а третий – что сумма цифр равна 30. Определить четырехзначный номер нарушителя.
Вопрос. Можно ли определить номер нарушителя, если показания дадут только два друга?

1.1 (29). Русская пирамида. Сколько кругов заданного радиуса r можно вырезать из правильного треугольника со стороной a?

1.1 (30). Счастливый год. Введите месяц и день своего рождения. Выясните, какой ближайший год будет для вас счастливым. Год называется счастливым, если остаток от деления суммы его цифр на 10 совпадает с аналогичным остатком сумм цифр месяца или дня рождения.

Задача 1.2

1.2 (1). С клавиатуры вводится целое число x из промежутка [100, 9999]. Если число четырехзначное, то найти сумму его цифр, а если трехзначное, то произведение цифр числа.

1.2 (2). Найти все натуральные числа, не превосходящие заданного m, двоичная запись которых представляет собой последовательность троек нулей и единиц (например: 111000111). Показать десятичную и двоичную запись этих чисел.

1.2 (3). Определить k-ю цифру последовательности
182764125216343… ,
в которой выписаны подряд кубы натуральных чисел.

1.2 (4). Найти все двузначные числа, сумма цифр которых не меняется при умножении числа на 2, 3, 4, 5, 6, 7, 8, 9.

1.2 (5). Найти все натуральные числа, не превосходящие заданного n, десятичная запись которых есть строго возрастающая или строго убывающая последовательность цифр.

1.2 (6). Найти сумму первых трех цифр дробной части вещественного числа.

1.2 (7). Дано натуральное k и последовательность цифр
100101102103104…
в которой выписаны все трехзначные числа. Определить k-ю цифру последовательности.

1.2 (8). Дано четырехзначное число. Найти число, образованное перестановкой двух первых и двух последних цифр заданного числа. Например, из числа 4566 получается 6645, из числа 7304 – 473.

1.2 (9). Составить программу, которая преобразует введенное с клавиатуры дробное число в денежный формат. Например, число 12,348 должно быть преобразовано в 12 руб. 35 коп.

1.2 (10). Даны натуральное n и вещественное f. Вычислить: $p = \prod_{i=1}^{n} \frac{1}{\sum_{k=0}^{i} (f + k)}$.

1.2 (11). Определить k-ю цифру последовательности
14916253649…
в которой выписаны подряд квадраты натуральных чисел.

1.2 (12). Найти все натуральные числа, не превосходящие заданного m, двоичная запись которых представляет собой последовательность единиц. Показать десятичную и двоичную запись этих чисел.

1.2 (13). Заданное натуральное число n, не превосходящее 100, записать прописью, например: 75 – «семьдесят пять».

1.2 (14). Натуральное число называется *совершенным*, если оно равно сумме всех своих делителей, меньших, чем оно само (например, 6 = 1 + 2 + 3). Найти все совершенные числа, не превосходящие заданного n.

1.2 (15). Числа Фибоначчи образуют бесконечную последовательность вида:
1, 1, 2, 3, 5, 8, 13, 21, 34, …
Определить номер последнего числа Фибоначчи, которое входит в диапазон **int**. Определить сумму первых чисел Фибоначчи, таких, что значение суммы не превышает этого диапазона.

1.2 (16). Перевести заданное целое число в систему римского счета.
Справка. Римские цифры обозначаются следующими латинскими буквами:

1 – I	50 – L	1000 – M ...
5 – V	100 – C	5000 – $\overline{V}$
10 – X	500 – D	10000 – $\overline{X}$

1.2 (17). Найти и распечатать все натуральные трехзначные числа, равные сумме кубов своих цифр.

1.2 (18). Напечатать столбиком пример на умножение двух заданных натуральных чисел k и l.

1.2 (19). Любая целочисленная денежная сумма $s > 7$ рублей может быть выдана без сдачи «трешками» и «пятерками». Найти для заданной суммы s необходимое количество «трешек» и «пятерок».

1.2 (20). Найти сумму цифр дробной и сумму цифр целой части вещественного числа. Вывести наибольшую сумму.

1.2 (21). Напечатать столбиком пример на деление натурального числа k на натуральное l (k делится на l нацело).

1.2 (22). Найти все натуральные числа, не превосходящие заданного m, двоичная запись которых представляет собой последовательность пар нулей и единиц (например: 11001100). Показать десятичную и двоичную запись этих чисел.

1.2 (23). Найти 10 первых троек пифагоровых чисел, то есть таких целых k, l, m, что $k^2 + l^2 = m^2$.

1.2 (24). Дано натуральное число n. Вычислить: $s = \sum_{i=1}^{n} \prod_{j=1}^{i} \frac{(j+i)!}{i!}$

1.2 (25). Найти все натуральные числа, не превосходящие заданного m, двоичная запись которых представляет собой симметричную последовательность нулей и единиц (начинающуюся с единицы). Показать десятичную и двоичную записи этих чисел.

1.2 (26). Найти все натуральные числа от 1 до 1000, которые совпадают с последними разрядами своих квадратов, например: $25^2 = 625$; $76^2 = 5676$.

1.2 (27). Найти натуральное число из заданного интервала, в двоичном представлении которого больше всего единиц. Вывести это число в двоичном и десятичном представлении.

1.2 (28). Определить k-ю цифру последовательности
112358132134… ,
в которой выписаны подряд все числа Фибоначчи.

1.2 (29). Дано натуральное n. Определить, упорядочены ли по возрастанию или по убыванию цифры в записи этого числа.

1.2 (30). Дано натуральное k и последовательность цифр
101112131415…
в которой выписаны все двузначные числа. Определить k-ю цифру последовательности.

2. Функции

функции, разработка пользовательского меню, рекурсия.

Контрольные вопросы

1. В чем заключается суть нисходящего метода программирования? Почему не написать всю программу внутри функции **main()**? Какими правилами вы пользуетесь, «разбивая» программу на функции?
2. Каковы наиболее распространенные ошибки при работе с функциями? С какими ошибками пришлось столкнуться вам?
3. Что такое прототип функции, каким целям он служит? Можно ли реализовать функцию без прототипа? В чем различие прототипа и заголовка функции?
4. Функции какого типа не возвращают значений? Значение какого типа возвращают функции, у которых возвращаемый тип не указан явно?
5. Какой оператор позволяет немедленно выйти из функции? Какое значение вернет следующая функция?

```
double Square(double x)
{
  double y;
  y = x * x;
}
```

6. Что такое локальные и глобальные переменные? Какую область видимости и время жизни они имеют? Где хранятся? Должны ли глобальные и локальные переменные иметь уникальные идентификаторы? Объясните, в каких случаях предпочтительнее использовать глобальные, а в каких – локальные переменные.
7. В чем состоит различие между параметрами и аргументами функции? Может ли быть аргументом функции константа, переменная, выражение, функция? Можно ли внутри функции изменить ее параметры и аргументы? Что такое безымянные параметры?
8. В каком заголовочном файле хранятся прототипы математических функций и математические константы? В чем различие между функциями **abs**() и **fabs**()? **ceil**() и **floor**()? **log**() и **log10**()? **tan**() и **atan**()?
9. Что такое рекурсия? Каковы плюсы и минусы ее использования? Объясните термины: «рекурсия сворачивается» и «рекурсия разворачивается». Всякую ли рекурсивную функцию можно реализовать без рекурсии?
10. Какие данные записываются и считываются из стека при выполнении следующего фрагмента?

```
double Square(double x)
```

```
{
  return x * x;
}

void main(void)
{
  double a = 4.0, b;
  b = Square(a);
}
```

11. Как с помощью математической функции **floor**() реализовать следующие функции?

```
RoundToTenths(number)          /* округление до десятых */
RoundToHundreths(number)       /* округление до сотых */
RoundToThousandths(number)     /* округление до тысячных */
```

12. Какие ошибки содержатся в следующей программе?

```
#include <stdio.h>

int Power(long base)
{
  int i; long temp = base;
  for (i = 0; i < n; i++)
    base *= temp;
  return base;
}
void main(void)
{
  const int n = 5;
  const long base = 10, result;
  result = Power(base);
  printf("Result = %ld", result);
}
```

13. Почему не работает функция **Calculator()?**

```
#include <stdio.h>

void Calculator(double cost, int last, int prev, double tax)
{
  int kwts = last – prev;
  cost = kwts * tax;
}
void main(void)
{
  int last = 1710, prev = 1605;
  double cost, tax = 54.6;
  /* Вычислить стоимость электроэнергии */
  Calculator(cost, last, prev, tax);
  printf("Cost = %lf", cost);
}
```

Примеры решения задач

Пример 2.1. Написать и протестировать функцию для вычисления x^n

ex21.c

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

double power(double x, int n);

void main(void)
{
  int n = -2;
  double x = 2.0;

  printf("-3 в степени 3 = %lg\n", power(-3.0, 3));
  printf("%lg в степени %d = %lg\n", x, n, power(x, n));
  printf("0 в степени -\n", power(0.0, -2));
}

double power(double x, int n)
{
  double step;

  if (n > 0) {
    for (step = 1.0; n > 0; n--) step *= x;
    return step;
  }
  else if (x != 0) {
    for (step = 1.0; n < 0; n++) step /= x;
    return step;
  }
  else {
    printf("Ошибка! ");
    return 0;
  }
}
```

Функция **power()** возвращает в вызывающую функцию значение с помощью оператора **return**. Значение, возвращаемое таким образом, может использоваться в выражениях.

Чтобы проверить правильность работы написанной функции, необходимо протестировать ее на нескольких типах входных данных. Тестовые данные надо подобрать таким образом, чтобы проследить работу всех ветвей составленного алгоритма. В нашем случае достаточно посмотреть, как работает функция вычисления степени на трех типах тестовых данных, что и сделано в функции **main()**.

Пример 2.2. Для заданной погрешности ε найти наименьшее n такое, что: $2^n / n! < \varepsilon$. Вывести все члены последовательности от 1-го до n-го.

Приведем вариант программы, использующий для решения задачи рекурсивную функцию:

ex22a.c

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

const double E = 0.001;
int sequence(int n); /* прототип рекурсивной функции */

void main(void)
{
  int min;

/* Запуск рекурсии. Переменной min будет присвоено наименьшее n,
   при котором выполняется условие задачи */
  min = sequence(1);
  printf("\n\nn = %d", min);
}

int sequence(int n)
{
  int i;
  double power = 1.0, fact = 1.0, term;

  for (i = 1; i <= n; i++) {
    power *= 2;          /* используя рекуррентные зависимости */
    fact *= i;           /* вычисляется 2^n и n! */
  }
  term = power / fact;
  printf("\n%02d %lg", n, term);
  if (term < E)          /* если условие выполняется, то */
     return n;           /* рекурсия "разворачивается" */
  return sequence(n+1); /* иначе рекурсивные вызовы функции */
}
```

Рекурсивная функция **sequence()** принимает параметр **n** – номер члена последовательности $2^n / n!$. Вначале его значение равно единице, т.к. поиск нужного члена последовательности мы будем вести, начиная с первого.

Внутри функции вычисляется значение n-го члена, которое сравнивается с константой **E**. Если условие задачи не выполняется, то функция **sequence()** вызывает саму себя с параметром, увеличенным на единицу, и вычисляется следующий член последовательности. В противном случае рекурсия начинает «разворачиваться», возвращая в точку вызова номер искомого члена последовательности.

Недостатком функции **sequence()** является то, что значения 2^n и $n!$ при каждом вызове функции вычисляются с самого начала. Их можно было бы

передавать как параметры функции, что значительно сократило бы объем вычислений. Кроме того, многократные вызовы рекурсивной функции исчерпывают память стека, что может привести к ошибке переполнения при задании очень высокой точности **E**. В этом смысле итерационные алгоритмы являются более быстрыми и эффективными.

Приведем еще один вариант программы, лишенный указанных недостатков:

ex22b.c

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

const double E = 0.001;

void main(void)
{
  int n = 0;
  double term, fact = 1.0, power = 1.0;

  do {
    n++;
    power *= 2;
    fact *= n;
    term = power / fact;
    printf("\n%02d %lg", n, term);
  } while (term > E);

  printf("\n\nn = %d", n);
}
```

Задачи

Требования к программам

1. Код программы должен быть логично разбит на функции.
2. Работа программы должна точно удовлетворять условиям задачи.
3. Программа не должна содержать предупреждений компилятора.
4. Программа должна быть хорошо протестирована.
5. Программа должна иметь дружественный интерфейс.
6. Код программы должен быть выдержан в хорошем стиле.

Указания к задаче 2.1

- Программу нужно реализовать в виде меню, для каждого пункта которого разрабатывается собственная функция;

- все действия пользователя должны обрабатываться программой, при вводе им некорректных данных должны выводиться соответствующие сообщения;
- до ввода пользователем корректных входных данных другие функции программы должны оставаться недоступными;
- пользователь должен иметь возможность ввести входные данные несколько раз без перезапуска программы;
- программа должна общаться с пользователем на понятном ему языке (русском, а если он не поддерживается, – на английском языке).

Указание к задаче 2.2

- Для решения задачи разработать две функции: рекурсивную и итерационную. Определить, какой из подходов эффективнее.

Задача 2.1

2.1 (1). Тетраэдр. Разработать программу, меню которой позволяет выполнить следующие функции:

1. Ввод длины ребра тетраэдра.
2. Вывод общей длины всех ребер тетраэдра.
3. Вывод площади всех граней тетраэдра.
4. Вывод объема тетраэдра.
5. Вывод высоты тетраэдра.
6. Вывод радиуса вписанного шара.
7. Информация о версии и авторе программы.
8. Выход из программы.

2.1 (2). Рациональные числа. Разработать программу, меню которой позволяет выполнить следующие функции:

1. Ввод (числитель и знаменатель) двух рациональных чисел.
2. Вывод двух рациональных чисел (в форме: $\frac{n}{m}$).
3. Сложение чисел.
4. Вычитание чисел.
5. Умножение чисел.
6. Деление чисел.
7. Получение гармонического значения двух чисел.
8. Информация о версии и авторе программы.
9. Выход из программы.

Справка: Гармоническое значение двух чисел получается путем получения обратных значений двух чисел, усреднения их и получения обратного значения результата.

2.1 (3). **Три отрезка.** Разработать программу, меню которой позволяет пользователю выполнить следующие функции:

1. Ввести координаты начала и конца трех отрезков в двумерном пространстве.
2. Определить, можно ли из этих отрезков составить треугольник.
3. Если это возможно, то определить вид треугольника (правильный, прямоугольный, равнобедренный, произвольного вида).
4. Вычислить периметр образованного треугольника.
5. Вычислить площадь треугольника
6. Информация о версии и авторе программы.
7. Выход из программы.

2.1 (4). **Дата.** Разработать программу, меню которой позволяет выполнить следующие функции:

1. Ввод даты (число, месяц, год).
2. Вывод даты в форме: **1 января 2010 г.**
3. Вывод даты в форме: **01.01.10.**
4. Вычисление по дате дня недели и порядкового номера дня в году.
5. Вывод количества дней, прошедших от Рождества Христова до введенной даты.
6. Информация о версии и авторе программы.
7. Выход из программы.

2.1 (5). **eVegetables.** Торговое предприятие, реализующее продукты питания, планирует открыть интернет-магазин по продаже овощей. Предприятие заказывает вам разработку программного модуля на C, который должен выполнять следующие функции:

1. Заказ картофеля (кг).
2. Заказ моркови (кг).
3. Заказ свеклы (кг).
4. Корзина (вывод сведений о количестве заказанных овощей и их стоимости).
5. Расчет стоимости заказа (сведения о стоимости каждого вида овощей, затратах на доставку, скидке на данный заказ, общей стоимости заказа).
6. Обратная связь (название магазина, номер лицензии, контактная информация).
7. Выход из программы.

Справочная информация. Предприятие продает картофель по цене 500 р. за кг, морковь – 1000 р. за кг и свеклу – 700 р. за кг. Стоимость доставки по городу составляет 15 000 р. eVegetables

предоставляет 10%-ную скидку на заказы весом более 10 кг, 20%-ную –более 25 кг и 30%-ную – более 50 кг.

2.1 (6). **Eurotrans Group.** Компания по грузоперевозкам заказывает вам разработку программы управления заказами, которая должна выполнять следующие действия:

1. Ввод веса груза (тонн).
2. Ввод расстояния перевозки (км).
3. Анализ данных и вывод решения о принятии заказа.
4. Расчет и вывод параметров заказа (рассчитывается в случае принятия заказа) – необходимое количество машин, стоимость страховки, общая стоимость заказа на перевозку.
5. Информация о версии и авторе программы.
6. Выход из программы.

Справочные сведения. Компания имеет 15 грузовых машин, каждая из которых может перевозить не более 20 тонн груза. Тариф составляет $2 за километр пробега. Компания считает экономически нецелесообразным принимать заказы на перевозку менее 50 тонн груза. Максимальное расстояние перевозки – 4000 километров. В стоимость заказа обычно включается страховка, составляющая 5% от общей стоимости.

2.1 (7). **Приорбанк.** Разработать программу, меню которой позволяет выполнить следующие функции:

1. Ввод денежной суммы в белорусских рублях.
2. Вывод аналогичной суммы в долларах США, евро и российских рублях.
3. Вывод таблицы курсов валют (покупка-продажа, взять данные на день разработки программы).
4. Расчет прибыли от операции для каждой валюты. Определение наиболее выгодной для банка валюты для сделки.
5. Краткая информация о банке и контакты.
6. Выход из программы.

2.1 (8). **Калькулятор.** Разработать программу, позволяющую пользователю рассчитать оптимальный тариф в сети МТС. Меню программы должно содержать следующие функции:

1. Ввод пользователем входных параметров (количество минут разговоров внутри сети / другим операторам / на городскую линию (в среднем в месяц), количество SMS).
2. Расчет оптимального для данного пользователя тарифа.
3. Информационные услуги (перечень существующих тарифов с описанием)
4. Обратная связь (контактная информация компании Velcom)

5. Выход из программы.

Примечание: информация о тарифах: www.mts.by

2.1 (9). **Меры длины.** Разработать программу, меню которой позволяет выполнить следующие функции:

1. Ввод значения длины (километры, метры, сантиметры, миллиметры).
2. Перевод длины в русские традиционные единицы (версты, аршины, сажени, вершки).
3. Перевод длины в английские традиционные единицы (мили, ярды, футы, дюймы).
4. Вывод отчета, представляющего введенное значение в 1) километрах, 2) метрах, 3) сантиметрах, 4) миллиметрах.
5. Информация о версии и авторе программы
6. Выход из программы.

Справка: Русские традиционные меры длины: 1 верста = 500 саженей; 1 сажень = 3 аршина; 1 аршин = 16 вершков; 1 вершок = 44,45 мм. Английские традиционные меры длины: 1 миля = 1760 ярдов; 1 ярд = 3 фута; 1 фут = 12 дюймов; 1 дюйм = 2,54 см.

2.1 (10). **Gross Transport.** Фирма, занимающаяся «перегонкой» автомобилей из Европы, заказывает вам программу, которая должна выполнять следующие действия:

1. Ввод стоимости автомобиля в стране Евросоюза.
2. Ввод года выпуска автомобиля.
3. Ввод объема двигателя.
4. Ввод расстояния прогона (км).
5. Расчет стоимости услуги (вывод стоимости перегона, стоимости растамаживания, общей цены автомобиля в РБ).
6. Информация о версии и авторе программы
7. Выход из программы.

Справочные сведения. Тариф фирмы составляет €0,5 за километр прогона. В конечную стоимость автомобиля также включается цена растамаживания, зависящая от объема двигателя и «возраста» автомобиля:

- 3 года и менее – €0,6 за 1 см.куб.;
- от 3 до 10 лет включительно: с рабочим объемом цилиндров двигателя до 2500 см.куб. – €0,35 за 1 см.куб.; с рабочим объемом цилиндров двигателя 2500 см.куб. и более – €0,6 за 1 см.куб.;
- от 10 до 14 лет – €0,6 за 1 см.куб.;
- 14 и более лет – €2 за 1 см.куб.

2.1 (11). **Комплексное число**. Разработать программу, меню которой позволяет выполнить следующие функции:

1. Ввод комплексного числа.
2. Вывод комплексного числа в алгебраической форме.
3. Вывод комплексного числа в показательной форме.
4. Получение сопряженного комплексного числа.
5. Возведение комплексного числа в целую положительную степень.
6. Информация о версии и авторе программы.
7. Выход из программы.

2.1 (12). Треугольник. Разработать программу, меню которой позволяет выполнить следующие функции:

1. Ввод координат вершин треугольника.
2. Определить вид треугольника (правильный, прямоугольный, равнобедренный, произвольного вида).
3. Вывод периметра треугольника.
4. Вывод площади треугольника.
5. Вычислить и вывести радиусы вписанной и описанной вокруг треугольника окружностей.
6. Информация о версии и авторе программы.
7. Выход из программы.

2.1 (13). Air Comfort. Компания занимается установкой стеклопакетов; планируется, что после открытия сайта компании, пользователи смогут спроектировать и рассчитать стоимость установки стеклопакетов прямо в интернете. Вам поручается разработать модуль на C, выполняющий следующие функции:

1. Ввод количества окон в квартире.
2. Ввод количества балконов.
3. Ввод этажа, на котором находится квартира.
4. Расчет и вывод параметров проекта: общая площадь застекления, стоимость застекления окон, стоимость установки балконной двери, процентная надбавка.
5. Вывод общей стоимости проекта.
6. Краткая информация о компании и контакты.
7. Выход из программы.

Справочные сведения. Стоимость одного стандартного оконного пакета (2,15×1,50 м) составляет $100, включая установку. Стоимость балконной двери (0,70×2,15 м) составляет $150, включая установку. Если квартира находится выше первого этажа, за установку каждого стеклопакета взимается дополнительно 15% его стоимости.

2.1 (14). Шар. Разработать программу, меню которой позволяет выполнить следующие функции:

1. Ввод радиуса шара.
2. Вывод площади поверхности шара.
3. Вывод объема шара.
4. Вывод длины ребра вписанного правильного тетраэдра.
5. Вывод площади сечения, проведенного на расстоянии k от центра шара.
6. Информация о версии и авторе программы.
7. Выход из программы.

2.1 (15). Project Manager. Софтверная компания Insoft заказывает вам программу, позволяющую управлять программными проектами внутри компании. Программа должны выполнять следующие действия:

1. Ввод даты начала и даты окончания проекта.
2. Ввод количества разработчиков.
3. Вывод длительности проекта (кол-во календарных и рабочих дней).
4. Расчет трудоемкости проекта (вывод кол-ва человеко-дней и человеко-часов).
5. Расчет финансовых параметров (общая стоимость проекта; сумма зарплаты программистам; накладные расходы; прибыль компании).
6. Принятие решение о целесообразности принятия проекта (указывается получаемая компанией прибыль).
7. Информация о версии и авторе программы.
8. Выход из программы.

Справочные сведения. Компания имеет почасовую ставку (rate) \$9. Накладные расходы составляют, как и в среднем по отрасли, 250% от заработной платы, выплачиваемой программистам. Средняя зарплата программистов, принятая в компании, – \$400. Рабочая неделя в компании составляет 40 часов (5 дней по 8 часов). Руководство компании считает нецелесообразным выполнение проектов, чистая прибыль по которым составляет менее \$1000.

2.1 (16). Алгебраические полиномы. Разработать программу, меню которой позволяет выполнить следующие функции:

1. Ввод двух полиномов 3 степени.
2. Вывод полиномов.
3. Сложение полиномов.
4. Вычитание полиномов.
5. Умножение полиномов.
6. Деление полиномов.
7. Информация о версии и авторе программы.
8. Выход из программы.

2.1 (17). Звездные войны. Звезда Epsilon 2 Gamma имеет три планеты: E2G_1, E2G_2 и E2G_3. Звездолету GExplorer предстоит встреча с другим кораблем в данной звездной системе. Астронавигаторы рассчитывают на появление звездолета в пространстве системы Epsilon 2 Gamma в момент противостояния трех планет (т.е. когда они выстроятся в одну линию по отношению к звезде). Командир корабля поручает вам разработать программный модуль, который должен выполнять следующие функции:

1. Ввод параметров планеты 2G_1 (масса, радиус орбиты, скорость движения)
2. Ввод аналогичных параметров планеты 2G_2.
3. Ввод аналогичных параметров планеты 2G_3.
4. Расчет продолжительности года на каждой из трех 3-х планет.
5. Рассчитать, на каком расстоянии друг от друга будут находиться планеты для заданного времени *t*, прошедшего с момента появления звездолета в системе.
6. Определить силу попарного гравитационного взаимодействия 3-х планет в заданное время *t*.
7. Выход из программы.

2.1 (18). Калькулятор. Разработать программу, позволяющую пользователю рассчитать оптимальный тариф в сети Velcom. Меню программы должно содержать следующие функции:

1. Ввод пользователем входных параметров (количество минут разговоров внутри сети / другим операторам / на городскую линию (в среднем в месяц), количество SMS).
2. Расчет оптимального для данного пользователя тарифа.
3. Информационные услуги (перечень существующих тарифов с описанием)
4. Обратная связь (контактная информация компании Velcom)
5. Выход из программы.

Примечание: информация о тарифах: www.velcom.by

2.1 (19). ТранзитБанк. Банк поручает вам разработать программу, которая должна выполнять следующие функции:

1. Открытие вклада (ввод суммы в белорусских рублях).
2. Дополнительный взнос на вклад (ввод суммы в белорусских рублях).
3. Просмотр остатка (суммы вклада).
4. Просмотр остатка на заданный день.
5. Закрытие вклада (печатается остаток вклада без процентов).
6. Краткая информация о банке и контакты.
7. Выход из программы.

Справочные сведения. Годовую процентную ставку по вкладу принять равной 12,75%. Проценты начисляются в первый день каждого месяца и суммируются с остатком вклада. Длительность вклада – 1 год, после этого срока проценты не начисляются.

2.1 (20). Круг. Разработать программу, меню которой позволяет выполнить следующие функции:

1. Ввод радиуса круга.
2. Вывод площади круга.
3. Вывод площади сектора круга, образованного углом *a*.
4. Вывод длины соответствующей окружности.
5. Вывод стороны квадрата, вокруг которого описана окружность.
6. Вывод объема конуса, основанием которого служит круг, а высота которого равна введенному радиусу.
7. Информация о версии и авторе программы.
8. Выход из программы.

2.1 (21). Capri Pizza. Вам необходимо разработать программу планирования закупок основных продуктов, которая должна выполнять следующие функции:

1. Ввод промежутка времени работы пиццерии (задаются начало и конец временного интервала – часы и минуты).
2. Вывод предполагаемого количества посетителей, которые посетят пиццерию в указанный промежуток времени.
3. Расчет количества ветчины, грибов, овощей и муки, необходимых для приготовления нужного количества пицц и вывод общей стоимости заказа.
4. Информационный раздел: стоимость пиццы с ветчиной, грибами и овощами.
5. Расчет прибыли Capri Pizza за указанный интервал времени.
6. Информация о версии и авторе программы.
7. Выход из программы.

Справочные сведения: В меню пиццерии: пиццы с ветчиной, с грибами и овощные пиццы, которые пользуются одинаковой популярностью. Для приготовления каждой из них необходимо соответственно: 90 гр. ветчины (закупочная стоимость – 7000 руб. за килограмм), 80 гр. грибов (10000 руб./кг) либо 150 гр. овощей (3000 руб./кг). Расход муки на пиццу любого типа – 150 гр. (1500 руб./кг). Стоимость приготовленной пиццы составляет 450% от себестоимости основных продуктов (в стоимость входит цена доп. продуктов, обслуживание, налоги и 10%-ая прибыль).

В утренние часы работы (9:00-13:00) пиццерию посещают в среднем 5 человек в час, днем (13:00-18:00) – 14 человек в час,

вечером (18:00-23:00) – 42 человека в час. Как правило, каждый заказывает одну пиццу.

2.1 (22). Гороскоп. Разработать программу, которая позволяет определить совместимость двух партнеров по гороскопу. Меню программы должно содержать следующие пункты:

1. Ввод данных первого партнера (число и месяц рождения, пол).
2. Ввод данных второго партнера (число и месяц рождения, пол).
3. Определение знака зодиака каждого партнера.
4. Решение, возможна ли между партнерами дружба.
5. Решение, подходят ли они друг другу в бизнесе.
6. Решение, смогут ли они жить в браке.
7. Информация о версии и авторе программы.
8. Выход из программы.

Справка. Астрологи полагают, что партнеры могут дружить, если их знаки зодиака принадлежат одной стихии (воздуха, огня, воды или земли). Партнеры подходят друг для друга в бизнесе, если имеют один знак зодиака либо смежные знаки. Партнеры имеют предпосылки для гармоничной семейной жизни, если они разного пола и их знаки зодиака принадлежат 1) стихиям воздуха и воды или 2) огня и земли.

2.1 (23). eFruit. Торговое предприятие планирует открыть интернет-магазин по продаже фруктов. Вам заказывают разработку программного модуля на C, который должен выполнять следующие функции:

1. Заказ мандаринов (кг).
2. Заказ персиков (кг).
3. Заказ винограда (кг).
4. Корзина (вывод сведений о количестве заказанных фруктов и их стоимости).
5. Расчет стоимости заказа (сведения о стоимости каждого вида фруктов, скидках на данный заказ, затратах на доставку, общей стоимости заказа).
6. Обратная связь (название магазина, номер лицензии, контактная информация).
7. Выход из программы.

Справочная информация. Предприятие продает мандарины по цене $1,14 за кг, персики – по цене $1,00 за кг и виноград – $1,28 за кг. eFruit предоставляет 10%-ную скидку за заказы стоимостью от $100, не считая затрат на доставку. eFruit берет $1 за доставку и обработку заказов весом менее 5 кг, $3 – за заказы весом от 5 до 20 килограмм. Если вес заказа превышает 20 кг – берется $10 плюс $2 за каждый килограмм.

2.1 (24). Компьютерные сети. Монтажная фирма AMP заказывает вам программу, которая должна выполнять следующие функции:

1. Ввод количества подключений.
2. Ввод количества этажей, которые хотят подключиться к сети.
3. Ввод высоты потолков и толщины перекрытий дома.
4. Ввод среднего расстояния от точки подключения до распределительного шкафа на этаже.
5. Расчет стоимости проектирования сети.
6. Расчет общей длины кабеля.
7. Расчет стоимости необходимого сетевого оборудования.
8. Расчет общей стоимости проекта.
9. Краткая информация о фирме и контакты.
10. Выход из программы.

Справочные сведения. Стоимость прокладки 1 метра кабеля составляет $1,5. Монтаж розетки в точке подключения – $1. Обжимка коннектора – $1 (на каждую точку подключения требуется 1 розетка и 2 коннектора). Монтаж шкафа на каждом этаже – $33 (стоимость самого шкафа – $110). Пробивка монтажного канала – $20. Проектирование сети обходится компании в 30% от суммарной стоимости оборудования и работ.

2.1 (25). Рациональное число. Разработать программу, меню которой позволяет выполнить следующие функции:

1. Ввод (числитель и знаменатель) рационального числа.
2. Вывод числа (в форме числителя и знаменателя, десятичной и «научной» нотации).
3. Вывод обратного к нему числа.
4. Вывод сокращенного рационального числа.
5. Информация о версии и авторе программы.
6. Выход из программы.

2.1 (26). Алгебраический полином. Разработать программу, меню которой позволяет выполнить следующие функции:

1. Ввод полинома 3 степени.
2. Вывод полинома.
3. Нормализация полинома.
4. Дифференцирование полинома.
5. Интегрирование полинома.
6. Вычисление значения полинома для заданного x.
7. Информация о версии и авторе программы.
8. Выход из программы.

2.1 (27). СтройИнвест. Предприятие по производству строительных материалов, имеющее филиалы в нескольких странах, заказывает

вам разработку программы управления заказами, которая должна выполнять следующие функции:

1. Заказ на цемент (тонн).
2. Заказ на гравий (тонн).
3. Заказ на бетон (тонн).
4. Выбор страны доставки.
5. Параметры заказа (вывод сведений о тарифах по каждому виду строительных материалов, о стоимости заказанных материалов, количестве вагонов, необходимых для перевозки, стоимости перевозки).
6. Решение о принятии заказа (в случае положительного решения выводится общая стоимость заказа с учетом доставки и прибыль предприятия).
7. Информация о версии и авторе программы
8. Выход из программы.

Справочные сведения. Предприятие реализует цемент по цене $230 за тонну (себестоимость составляет $160), гравий по цене $142 ($96), бетон – $260 ($205). Доставка осуществляется по железной дороге в товарных вагонах, каждый из которых грузится до 50 тонн. Оплачивает доставку предприятие-изготовитель. По Беларуси стоимость доставки составляет $200 за вагон, в Россию и Украину – $320 за вагон, в Молдавию – $360. Заказы из других стран не принимаются. Руководство завода считает экономически нецелесообразными заказы, прибыль от которых не превышает $25,000.

2.1 (28). Абитуриент. Разработать программу, меню которой позволяет выполнить следующие функции:

1. Вывод информации о факультете КСиС.
2. Вывод информации о факультете ИТиУ.
3. Вывод информации о факультете РЭ.
4. Вывод информации о факультете ТК.
5. Вывод информации о факультете КП.
6. Рекомендации. Программа запрашивает у абитуриента оценки по математике, физике, языку и средний балл аттестата. После этого выводится рекомендации, на какой факультет поступать.
7. Выход из программы.

Примечание: информация о факультетах и проходных баллах: http://abitur.bsuir.by/. Если баллы «не проходные» – рекомендация не тратить время, если балл по языку намного выше, чем по математике и физике, – рекомендовать поступать в гуманитарный вуз.

2.1 (29). Ремонт. Разработать программу, которая по введенным параметрам рассчитывает стоимость оклейки обой в комнате. Программа должна содержать следующее меню:

1. Ввод длины и ширины комнаты (м).
2. Ввод количества окон.
3. Ввод количества дверных проемов.
4. Ввод стоимости обойной трубки ($).
5. Расчет площади оклейки (кв. м).
6. Расчет необходимого количества обойных трубок.
7. Расчет стоимости поклейки (с учетом клея).
8. Выход из программы.

Справочные сведения. Окна имеют стандартные размеры 2,15×1,50 м, дверные проемы – 0,90×2,05 м, высоту потолков принять – 2,60 м. Ширина обойной трубки составляет 50 см. Стоимость 1 пакета обойного клея равна $2,5, его хватает на 5 трубок.

2.1 (30). Натуральное число. Разработать программу, меню которой позволяет выполнить следующие функции:

1. Ввод натурального числа.
2. Вывод значения числа в десятичной, восьмеричной и шестнадцатеричной форме.
3. Вывод обратного к нему числа в десятичной форме.
4. Перевод числа в систему счисления, с основанием n [2..16].
5. Информация о версии и авторе программы.
6. Выход из программы.

Замечание: для перевода числа в системы с другим основанием разработать собственные функции.

Задача 2.2

2.2 (1). Вычислить $x = \sqrt[3]{a}$ для заданного значения a, используя рекуррентное соотношение:

$$x_{n+1} = \frac{1}{3}\left(x_n + 2\sqrt{\frac{a}{x_n}}\right); \quad x_0 = a.$$

Сколько итераций надо выполнить для достижения заданной погрешности ε, т.е. чтобы $\left|x_{n+1} - x_n\right| \le \varepsilon$?

2.2 (2). Пусть $a_1 = 1 - \cos(x), \quad b_1 = x - \sin(x).$

Заданы рекуррентные соотношения:

$$a_i = a_{i-1} + (-1)^{i-1}\frac{x^{2i-2}}{(2i-2)!};\quad b_i = b_{i-1} + (-1)^{i-1}\frac{x^{2i-1}}{(2i-1)!},\quad i = 2,3,...$$

Испытать разложения на сходимость при разных значениях x. Сколько итераций потребуется выполнить, чтобы для заданной погрешности ε было выполнено соотношение: $|a_i - b_i| \le \varepsilon$?

2.2 (3). Пусть $a_1 = x - \sin(x),\quad b_1 = x - \dfrac{e^x - e^{-x}}{2}$.

Заданы рекуррентные соотношения:

$$a_i = a_{i-1} + (-1)^{i-1}\frac{x^{2i-1}}{(2i-1)!};\quad b_i = b_{i-1} + \frac{x^{2i-1}}{(2i-1)!},\quad i = 2,3,...$$

Испытать разложения на сходимость при разных значениях x. Сколько итераций потребуется выполнить, чтобы для заданной погрешности ε было выполнено соотношение: $|a_i - b_i| \le \varepsilon$?

2.2 (4). Заданы рекуррентные соотношения:

$$a_i = \frac{x^{2i}b_{i-1}}{i!};\quad b_i = \frac{(a_{i-1} + b_{i-1})}{x^2 i^2},\quad a_1 = b_1 = 10^5,\ i = 2,3,...$$

Испытать разложения на сходимость при разных значениях x. Сколько итераций потребуется выполнить, чтобы для заданной погрешности ε было выполнило соотношение: $|a_i - b_i| \le \varepsilon$?

2.2 (5). Разработать рекурсивный и итерационный вариант функции, которая вычисляет для заданного n сумму n первых чисел Фибоначчи:
1, 1, 2, 3, 5, 8, 13, …

2.2 (6). $$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - ... + (-1)^{n-1}\frac{x^{2n-1}}{(2n-1)!} + ...$$

Численно убедиться в справедливости равенства, для чего для заданного значения x вычислить его левую часть и разложение, стоящее в правой части. При каком n исследуемое выражение отличается от sin x менее, чем на заданную погрешность ε? Испытать разложение на сходимость при разных значениях x.

2.2 (7). $$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - ... + (-1)^n\frac{x^{2n}}{(2n)!} + ...$$

Численно убедиться в справедливости равенства, для чего для заданного значения x вычислить его левую часть и разложение, стоящее в правой части. При каком n исследуемое выражение

отличается от cos x менее, чем на заданную погрешность ε? Испытать разложение на сходимость при разных значениях x.

2.2 (8). Для заданного $x > 1$ вычислить $y = \sqrt{x}$ по итерационной формуле: $y_i = \frac{1}{2}\left(y_{i-1} + \frac{x}{y_{i-1}} \right)$ с заданной погрешностью ε, задав начальное приближение $y_0 = x$. Сравнить с результатом использования встроенной функции. Сколько итераций пришлось выполнить?

2.2 (9). $$\frac{e^x + e^{-x}}{2} = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \ldots + \frac{x^{2n}}{(2n)!} + \ldots$$

Численно убедиться в справедливости равенства, для чего для заданного значения x вычислить его левую часть и разложение, стоящее в правой части. При каком n исследуемое выражение отличается от левой части менее, чем на заданную погрешность ε? Испытать разложение на сходимость при разных значениях x.

2.2 (10). Для заданных a и p вычислить $x = \sqrt[p]{a}$, используя рекуррентную формулу

$$x_{n+1} = \frac{x_n}{p^2}\left[\left(p^2 - 1\right) + \frac{1}{2}(p+1)\frac{a}{x_n^p} - \frac{1}{2}(p-1)\frac{x_n^p}{a} \right].$$

Сколько итераций надо выполнить, чтобы для заданной погрешности ε было справедливо соотношение $\left| x_{n+1} - x_n \right| \le \varepsilon$? При каких начальных приближениях x_0 и a процесс сходится?

2.2 (11). $$\frac{e^x - e^{-x}}{2} = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots + \frac{x^{2n+1}}{(2n+1)!} + \ldots$$

Численно убедиться в справедливости равенства, для чего для заданного значения x вычислить его левую часть и разложение, стоящее в правой части. При каком n исследуемое выражение отличается от левой части менее, чем на заданную погрешность ε? Испытать разложение на сходимость при разных значениях x.

2.2 (12). Для заданных a и p вычислить $x = \sqrt[p]{a}$ по рекуррентному соотношению Ньютона:

$$x_{n+1} = \frac{1}{p}\left[(p-1)x_n + \frac{a}{x_n^{p-1}} \right]; \quad x_0 = a\,.$$

Сколько итераций надо выполнить, чтобы для заданной погрешности ε выполнялось соотношение $\left| x_{n+1} - x_n \right| \le \varepsilon$?

2.2 (13). $\ln x = 2\left[\frac{x-1}{x+1} + \frac{1}{3}\left(\frac{x-1}{x+2}\right)^3 + ... + \frac{1}{2n-1}\left(\frac{x-1}{x+1}\right)^{2n-1} + ...\right],\ x > 0$

Численно убедиться в справедливости равенства, для чего для заданного значения x вычислить его левую часть и разложение, стоящее в правой части. При каком n исследуемое выражение отличается от $\ln x$ менее, чем на заданную погрешность ε? Испытать разложение на сходимость при разных значениях x.

2.2 (14). $a^x = 1 + \frac{x \ln a}{1!} + \frac{(x \ln a)^2}{2!} + ... + \frac{(x \ln a)^n}{n!} + ...$

Численно убедиться в справедливости равенства, для чего для заданного значения x вычислить его левую часть и разложение, стоящее в правой части. При каком n исследуемое выражение отличается от a^x менее, чем на заданную погрешность ε? Испытать разложение на сходимость при разных значениях x.

2.2 (15). Заданы рекуррентные соотношения:

$x_i = x_{i-1} - \frac{y_{i-1}}{i^2};\quad y_i = y_{i-1} - \frac{x_{i-1}}{i^2},\quad i = 2, 3, ...;\ x_1 = 1,\ y_1 = 0.15.$

При каком i будет выполнено условие $|x_i - y_i| \le \varepsilon$, где $\varepsilon = 10^{-5}$?

2.2 (16). Пусть $a_1 = 1 - \cos(x),\quad b_1 = 1 - \frac{e^x + e^{-x}}{2}.$

Заданы рекуррентные соотношения:

$a_i = a_{i-1} + (-1)^{i-1}\frac{x^{2i-2}}{(2i-2)!};\quad b_i = b_{i-1} + \frac{x^{2i-2}}{(2i-2)!},\quad i = 2, 3, ...$

Испытать разложения на сходимость при разных значениях x. Сколько итераций потребуется выполнить, чтобы для заданной погрешности ε было выполнено соотношение: $|a_i - b_i| \le \varepsilon$?

2.2 (17). Для заданного x вычислить значение и номер минимального положительного члена числовой последовательности, заданной рекуррентным соотношением:

$a_i = a_{i-1} + \frac{i!}{a_{i-1}^4 x^i};\quad a_1 = -100;\quad i = 2,3,...$

2.2 (18). $\frac{1}{4}\ln\frac{1+x}{1-x} + \frac{1}{2}arctg\, x = x + \frac{x^5}{5} + ... + \frac{x^{4n+1}}{4n+1} + ...,\quad -1 < x < 1.$

Численно убедиться в справедливости равенства, для чего для заданного значения x вычислить его левую часть и разложение, стоящее в правой части. При каком n исследуемое выражение отличается от выражения в левой части менее, чем на заданную погрешность ε? Испытать разложение на сходимость при разных значениях x.

2.2 (19). $e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + ... + \frac{x^n}{n!} + ...$

Численно убедиться в справедливости равенства, для чего для заданного значения x вычислить его левую часть и разложение, стоящее в правой части. При каком n исследуемое выражение отличается от e^x менее, чем на заданную погрешность ε? Испытать разложение на сходимость при разных значениях x.

2.2 (20). $\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} - ... + (-1)^n \frac{x^{2n+1}}{2n+1} + ...$

Численно убедиться в справедливости равенства, для чего для заданного значения x вычислить его левую часть и разложение, стоящее в правой части. При каком n исследуемое выражение отличается от arctan x менее, чем на заданную погрешность ε? Испытать разложение на сходимость при разных значениях x.

2.2 (21). $(1 + 2x^2)e^{x^2} = 1 + 3x^2 + ... + \frac{2n+1}{n!}x^{2n} + ...$

Численно убедиться в справедливости равенства, для чего для заданного значения x вычислить его левую часть и разложение, стоящее в правой части. При каком n исследуемое выражение отличается от выражения в левой части менее, чем на заданную погрешность ε? Испытать разложение на сходимость при разных значениях x.

2.2 (22). $\ln(1 - x) = -\left(x + \frac{x^2}{2} + \frac{x^3}{3} + ... + \frac{x^n}{n} + ... \right), \; x < 1$

Численно убедиться в справедливости равенства, для чего для заданного значения x вычислить его левую часть и разложение, стоящее в правой части. При каком n исследуемое выражение отличается от $\ln(1–x)$ менее, чем на заданную погрешность ε? Испытать разложение на сходимость при разных значениях x.

2.2 (23). Заданы рекуррентные соотношения:

$$a_i = \frac{x^i b_{i-1}}{i!};\quad b_i = \frac{a_{i-1}}{x+i},\quad a_1 = 10^4, b_1 = 10^6, i = 2, 3, \ldots$$

Испытать разложения на сходимость при разных значениях x. Сколько итераций потребуется выполнить, чтобы для заданной погрешности ε было выполнено соотношение: $|a_i - b_i| \le \varepsilon$?

2.2 (24). Бином Ньютона. Для заданных m и x вычислить бином Ньютона $(1+x)^m$ непосредственно и по формуле разложения в ряд:

$$(1+x)^m = \sum_{i=0}^{m} C_m^i x^i \ .$$

Для вычисления C_m^i использовать рекуррентное соотношение:

$$C_m^{i+1} = C_m^i \frac{m-i}{i+1};\quad C_m^o = 1,$$

и классическую формулу $C_m^n = \dfrac{m!}{n!(m-n)!}$.

Какой из подходов эффективнее?

2.2 (25). $\pi = 3 + 4\left(\dfrac{1}{2\cdot 3\cdot 4} - \dfrac{1}{4\cdot 5\cdot 6} + \dfrac{1}{6\cdot 7\cdot 8} - \mathbf{K}\right)$.

Численно убедиться в справедливости равенства. При каком n выражение, стоящее в правой части, отличается от π менее, чем на заданную погрешность ε?

2.2 (26). $\pi = \sqrt{6\left(1 + \dfrac{1}{2^2} + \dfrac{1}{3^2} + \dfrac{1}{4^2} + \mathbf{K}\right)}$.

Численно убедиться в справедливости равенства. При каком n выражение, стоящее в правой части, отличается от π менее, чем на заданную погрешность ε?

2.2 (27). Число сочетаний. Для заданных m и n вычислить число сочетаний C_m^n непосредственно: $C_m^n = \dfrac{m!}{n!(m-n)!}$ и по рекуррентным формулам:

$$C_m^n = \frac{m-n+1}{n} C_m^{n-1},\quad C_m^1 = m;$$

$$C_m^n = C_{m-1}^{n-1} + C_{m-1}^n;\quad C_m^m = 1.$$

Указание: сравнить число операций по каждой формуле, какой из подходов эффективнее?

2.2 (28). Разработать рекурсивный и итерационный варианты функции, которая возвращает наибольший общий делитель (НОД) двух целых положительных чисел n и m, с помощью алгоритма Евклида: пусть m больше либо равно n. Если $n = 0$, то НОД(m, n) = m. Если n не равно нулю, то для чисел n, m и r, где r – остаток от деления m на n, выполняется равенство НОД(m, n) = НОД(n, r). Например, НОД(15, 6) = НОД(6, 3) = НОД(3, 0) = 3.

2.2 (29). Заданы рекуррентные соотношения:

$$a_i = \frac{1}{3}\left(a_{i-1} + 2\sqrt{\frac{x}{a_{i-1}}}\right); \quad b_i = \frac{1}{2}\left(b_{i-1} + \frac{x}{b_{i-1}}\right), \quad a_1 = b_1 = 1,\ x = 64,\ i = 2, 3, \ldots$$

Сколько итераций потребуется выполнить, чтобы для заданной погрешности $\varepsilon = 10^{-10}$ было выполнено соотношение: $|2a_i - b_i| \le \varepsilon$?

2.2 (30). $\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \mathbf{K}\right)$.

Численно убедиться в справедливости равенства, для чего вычислить выражение, стоящее в правой части. При каком n исследуемое выражение отличается от π менее, чем на заданную погрешность ε?

3. Массивы и указатели

одномерные и многомерные массивы, указатели, динамические массивы.

Контрольные вопросы

1. Что такое массив? Для чего в программировании нужны массивы?
2. Сколько элементов содержит массив **int data[50]**? Являются ли правильными обращения: **data[0]**, **data[50]**, **data[51]**?
3. Сколько способов инициализации трехмерного массива **cube[3][3][3]** вы знаете?
4. Можно ли массив, являющийся параметром функции, изменить внутри функции? Напишите заголовок функции, которой передается двумерный массив.
5. Что такое указатель? В чем различие указателя и имени массива? В чем разница между нулевыми указателями и указателями типа **void**?
6. Какие арифметические операции допускаются с указателями? Можно ли присваивать указатели друг другу? Использовать с указателями индексные выражения?
7. Какие функции позволяют выделять и освободить память в куче? Почему необходимо освобождать память в куче, после того, как динамические переменные уже не нужны?
8. Объясните следующие понятия: рандомизация программы, генерация псевдослучайных чисел. Какие функции C позволяют выполнить эти действия?
9. Какие параметры влияют на скорость сортировки? Оцените время работы методов «пузырька», вставок и быстрой сортировки. В каких случаях их целесообразно применять?
10. Объясните разницу между линейным и бинарным поиском. В каких случаях их используют? Сколько операций сравнений в среднем необходимо выполнить?
11. Какие ошибки содержат следующие фрагменты программы?

    1)
    ```
    #define SIZE 100;
    SIZE = 10;
    ```
    2)
    ```
    int b[10] = {0}, i;
    for (i = 0; i <= 10; i++)
      b[i] = 1;
    ```
    3)
    ```
    int a[2][2] = {1, 2}, {3, 4};
    a[1,1] = 5;
    ```
    4)
    ```
    char str[5];
    scanf("%s", str); /* пользователь вводит слово Hello */
    ```
    5)
    ```
    int *number;
    ```

```
printf("%d\n", *number);
```

6)
```
int *x, y;
x = y;
```

12. Можете ли вы написать однострочные операторы, которые присваивают целые случайные числа переменной *n* в следующих диапазонах:
    1) $1 \le n \le 2$
    2) $0 \le n \le 100$
    3) $1000 \le n \le 1112$
    4) $-3 \le n \le 11$

а также из следующих наборов чисел:

    5) 2, 4, 6, 8, 10
    6) 3, 5, 7, 9, 11
    7) 6, 10, 14, 18, 22?

13. Какой метод сортировки реализует данная функция?

```
void Sort(int a[], int n)
{
  int i, j, t;
  for (i = 0; i < n; i++)
    for (j = 0; j < n–i–1; j++)
      if(a[j] > a[j+1]) {
         t = a[j]; a[j] = a[j+1]; a[j+1] = t;
      }
}
```

14. Какой метод сортировки реализует данная функция?

```
void Sort(int a[], int n)
{
  int i, j, t;

  for (i = 1; i <= n-1; i++) {
    j = i;
    while (a[j] < a[j-1] && j >= 1) {
      t = a[j]; a[j] = a[j-1]; a[j-1] = t;
      j--;
    }
  }
}
```

Примеры решения задач

Пример 3.1. Сформировать массив натуральных чисел *a(n)* случайным образом. Распечатать те элементы массива, значения которых являются степенями двойки.

ex31.c

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

```
#include <time.h>
#define N 25   /* количество элементов в массиве */
#define M 32   /* диапазон случайных значений */

int IsPower2(int value);

void main(void)
{
  int i, a[N];

  puts("\nИсходный массив:\n");
  randomize();
  for (i = 0; i < N; i++) { /* инициализация массива */
    a[i] = random(M)+1;     /* случайными числами из [1..M] */
    printf("%d ", a[i]);
  }
  puts("\n\nИскомые элементы массива:\n");
  for (i = 0; i < N; i++)
    if (IsPower2(a[i]))
      printf("A[%d] = %d\n", i+1, a[i]);
}

int IsPower2(int value)
{
  int rem;

  if (value < 1)
    return 0;
  while (value > 1) {
    rem = value % 2;
    if (rem) return 0;
    value /= 2;
  }
  return 1;
}
```

Функция **IsPower2()** вычисляется остаток от деления переменной **value** на **2**. Если остаток не равен нулю (т.е. число нечетное), возвращается 0 (ложь). После этого **value** делится на 2, и снова вычисляется остаток. Число является степенью двойки, если все его остатки при делении равны нулю.

Пример 3.2. Сформировать матрицу $a(10,10)$ следующего вида:

$$\begin{pmatrix} 1 & 0 & 0 & . & . & 0 \\ 2 & 1 & 0 & . & . & 0 \\ 3 & 2 & 1 & . & . & 0 \\ . & . & . & . & . & 0 \\ . & . & . & . & . & 0 \\ 10 & 9 & 8 & . & . & 1 \end{pmatrix}$$

ex32.c

```
#include <stdio.h>
#define N 10

void main(void)
{
  int a[N][N];
  int i, j;

  for (i = 0; i < N; i++)
    for (j = 0; j < N; j++)
      if (i >= j) /* условие попадания под главную диагональ */
        a[i][j] = i - j + 1;
      else a[i][j] = 0;
  /* Вывод матрицы */
  for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++)
      printf("%3d", a[i][j]);
    putchar('\n');
  }
}
```

Задачи

Требования к программам

1. Код программы должен быть логично разбит на функции.
2. Работа программы должна точно удовлетворять условиям задачи.
3. Программа не должна содержать предупреждений компилятора.
4. Программа должна быть хорошо протестирована.
5. Программа должна иметь дружественный интерфейс.
6. Код программы должен быть выдержан в хорошем стиле.

Указание к задачам 3.1 и 3.2

- В программах использовать динамические массивы.

Задача 3.1

3.1 (1). Дан неупорядоченный целочисленный массив $a(n)$. Разработать функции сложения двух элементов массива и сравнения сумм. Других операций над массивом производить не разрешено. Найти наибольший элемент массива.

3.1 (2). Дана позиция элемента m в упорядоченном массиве $a(n)$, и количество допустимых операций l для его бинарного поиска. Определить, можно ли это сделать для заданного значения n.

3.1 (3). Система наблюдения. В массиве $h(n)$ хранятся значения высот некоторого профиля местности (ее вертикального сечения) с постоянным шагом по горизонтали. Найти области (номера точек измерения высоты), невидимые для наблюдателя, находящегося в точке h_0.

3.1 (4). Дано многозначное число. Вывести на печать число, состоящее из цифр, которые встречаются более одного раза в записи первоначального числа.

3.1 (5). Колокол. В массиве $a(n)$, сформированном случайным образом, наименьший элемент поместить на первое место, наименьший из оставшихся – на последнее место, следующий по величине – на второе место, следующий – на предпоследнее и так далее – до середины массива. Вывести исходный и сформированный массивы.

3.1 (6). Автостоп. Из пункта A в пункт B, между которыми s км, выехал студент на велосипеде с постоянной скоростью v_0 км/ч. Одновременно с ним в том же направлении другой студент решил добраться «автостопом» – на разных видах попутного транспорта. Перед каждым участком пути он τ_i минут «голосует», затем движется t_i часов со скоростью v_i км/ч (величины τ_i, t_i, v_i, $i = 1, \ldots, n$ заданы в соответствующих массивах). В каких точках пути (в какие моменты времени) друзья смогут помахать друг другу рукой?

3.1 (7). В заданном случайным образом массиве $a(n)$ найти i и j такие, что $\sum_{k=i}^{j} a_k$ максимально.

3.1 (8). Медиана. В массиве $a(2n+1)$, не содержащем одинаковых элементов, найти средний по величине элемент, то есть такой, что в массиве a ровно n элементов меньше его и столько же элементов больше его. Массив a сохранить (не сортировать), дополнительных массивов не использовать.

3.1 (9). Дан массив чисел. С помощью рекурсивной функции все числа, граничащие с цифрой 1, заменить нулями.

3.1 (10). Перемешать массив $a(n)$ случайным образом так, чтобы ни один из элементов гарантированно не остался на своем прежнем месте и имел бы одинаковые шансы занять любое из остальных мест.

3.1 (11). Задача Иосифа. По кругу располагаются n человек. Ведущий считает по кругу, начиная с первого, и выводит («казнит») m-го человека. Круг смыкается, счет возобновляется со следующего после «казненного»; так продолжается, пока «в живых» останется только один человек. Найти номер оставшегося «в живых» человека, а также для заданного n найти такое $m > 1$, при котором «в живых» останется первый.

3.1 (12). Пусть значение функции $f(n)$ равно количеству символов в русской записи количественного числительного n: $f(1) = 4$ (один), $f(3) = 3$ (три), $f(42) = 9$ (сорок два) и т.д. Найти все натуральные n, для которых $f(n) = n$.

3.1 (13). Вычислить факториал натурального числа n ($n > 13$). Т.к. целые типы данных применимы в ограниченном диапазоне натуральных чисел, для представления факториала использовать массив.
Рекомендация. В элементы целочисленного массива заносятся цифры числа (факториала). Первый элемент массива хранит количество цифр.

3.1 (14). Разработать рекурсивную функцию, которая в массиве $a(n)$, сформированном случайным образом, каждый элемент, кроме первого, заменяет суммой всех предыдущих элементов.

3.1 (15). Дан массив целых чисел и число q. Переставить элементы массива так, чтобы сначала были записаны числа, меньшие q, потом равные q, а в конце большие q. Дополнительные массивы не использовать.

3.1 (16). Для заданного n получить все возможные перестановки чисел: $1, 2, \ldots, n$.

3.1 (17). Новобранцы. Новобранцы выстроены в шеренгу. По команде НАЛЕ–ВО! одни из них поворачиваются налево, а другие – направо. Это есть начальное положение шеренги. Далее каждый новобранец действует следующим образом: увидев перед собой лицо соседа, он ровно через секунду поворачивается кругом. В конце концов повороты прекратятся. Промоделировать повороты новобранцев.

3.1 (18). В массиве $p(n)$ найти самую длинную последовательность, которая является арифметической или геометрической прогрессией.

3.1 (19). Имитировать перетасовку новой колоды игральных карт в 52 листа многократным применением операций сдвига и «врезки» так, чтобы никакие две рядом лежащие карты не сохранили бы свой первоначальный порядок.
Примечание. Операция врезки состоит в делении колоды на две неравные подколоды и слиянии частей случайного объема, взятых из обеих подколод.

3.1 (20). Тренажер по арифметике. Пользователь вводит разрядность операндов, тип операции: + – * / (на множестве натуральных чисел) и количество примеров. Компьютер генерирует случайным образом операнды, результат операции и выводит ученику серию примеров, в каждом из которых один из операндов или результат «замаскирован», например: 37 * __ = 1591. Ученик вводит пропущенное число (в приведенном примере – 41); компьютер проверяет правильность и ведет статистику ошибок.

3.1 (21). Дан набор чисел 1, 2, ..., n. Написать программу, которая для произвольной последовательности, состоящей из этих чисел, найдет наименьшее количество перестановок, необходимых для того, чтобы расположить числа в порядке возрастания.
Примечание. Под перестановкой понимается обмен местами любых двух чисел.

3.1 (22). Вычислить степень числа a^n, например, 3^{100}. Т.к. целые типы данных применимы в ограниченном диапазоне натуральных чисел, для представления a^n использовать массив.
Рекомендация. В элементы целочисленного массива заносятся цифры числа. Первый элемент массива хранит количество цифр в нем.

3.1 (23). В целочисленном массиве $k(n)$, заданном случайным образом, много повторяющихся элементов. Найти (в процентах) частоту появления каждого из m наиболее часто встречающихся элементов ($m << n$). Удалить все повторяющиеся элементы, оставив в массиве только один.

3.1 (24). Среди элементов массива $z(m)$ (сформированных случайным образом) найти k ($k << m$) наибольших. Поиск осуществить за один проход (просмотр) массива.

3.1 (25). Задан массив, состоящий из n неотрицательных случайно сформированных чисел. Найти в нем индекс элемента, для которого

сумма элементов, стоящих до него, наименее отличается от суммы элементов, стоящих после него.

3.1 (26). Циклический сдвиг. Осуществить циклический сдвиг элементов массива $t(n)$ на m позиций влево, то есть получить массив: $t_{m+1},...,t_n,t_1,...,t_m$. При этом необязательно $m < n$.

3.1 (27). Гистограмма. Для выборки случайных чисел, содержащейся в массиве $a(n)$, построить гистограмму распределения, содержащую m интервалов.

Справка. Гистограмма распределения основывается на массиве данных из m элементов, значение каждого из которых равно количеству (или доле из общего числа) чисел из a, попавших в соответствующий интервал. Ширина интервала определяется как $(a_{\max}–a_{\min}) / m$.

3.1 (28). Разработать рекурсивную функцию бинарного поиска заданного элемента в целочисленном массиве.

3.1 (29). Дано многозначное число. Вывести на печать число, состоящее из цифр, которые не встречаются в записи первоначального числа.

3.1 (30). Счастливый номер. Распечатать все «счастливые» номера от 0 до 999999. Номер является «счастливым», если сумма первых трех цифр равна сумме трех его последних цифр. Если в числе меньше 6 цифр, то недостающие начальные цифры считаются нулями.

Задача 3.2

3.2 (1). Ранг матрицы. Найти ранг прямоугольной матрицы $a(m, n)$ методом Гаусса.

3.2 (2). Динамический двумерный массив $a(n, n\text{-}1)$ заполнить случайными числами. После строки, содержащей максимальный элемент массива, вставить строку из нулей. Зеркально отобразить элементы массива относительно главной диагонали, а затем относительно побочной диагонали. Вывести на экран исходный и результирующий массивы.

3.2 (3). Шифрование. В организации принят следующий метод шифрования: сообщение записывают в квадратную таблицу подходящего размера, начиная с верхнего левого угла против часовой стрелки по спирали к центру. Затем оно считывается по

строкам и передается адресату. Написать программу шифрования и дешифрования сообщений указанным методом.

3.2 (4). Похожие строки. Два столбца матрицы назовем похожими, если они отличаются только порядком элементов. Найти все похожие столбцы.

3.2 (5). Король и ферзи. На шахматной доске находятся король и несколько ферзей другого цвета. Проверить, находится ли король под угрозой и если да, кто ему угрожает. Положение фигур задано массивом $k(8, 8)$: 0 – клетка пуста, 1 – король, 2 – ферзь.

3.2 (6). Морской бой. Напечатать заготовку для игры в «морской бой», то есть расположить случайным образом 10 «линейных» кораблей (1 – 4-палубный, 2 – 3-палубных, 3 – 2-палубных, 4 – 1-однопалубных) на поле размером 10×10 так, чтобы они не касались друг друга.

3.2 (7). Ввод элементов матрицы $a(m, n)$ осуществляется в произвольном порядке тройками чисел $< i, j, a_{ij} >$. Признаком конца ввода служат три нуля: <0,0,0>. Проверить корректность такого ввода: все ли элементы введены, нет ли попытки повторного вода или указания несуществующих координат i и j.
Указание. При попытке повторного ввода пользователю должен предоставляться выбор: оставить старое значение или заменить его новым. По завершении ввода матрица распечатывается; при этом невведенные элементы замещаются символами '*'.

3.2 (8). Матрицу $a(m, n)$ заполнить следующим образом. Для заданных k и l элементу a_{kl} присвоить значение 1; элементам, окаймляющим его (соседним с ним по вертикали, горизонтали и диагонали) – значение 2; элементам следующего окаймления – значение 3 и так далее до заполнения всей матрицы.
Примечание. Алгоритм не изменится, если координаты элемента (несуществующего) k и l находятся за пределами матрицы.

3.2 (9). Замочная скважина. Даны мозаичные изображения замочной скважины и ключа. Пройдет ли ключ в скважину? Т. е. даны матрицы $k(m_1, n_1)$ и $l(m_2, n_2)$, $m_1 > m_2$, $n_1 > n_2$, состоящие из нулей и единиц. Проверить, можно ли наложить матрицу l на матрицу k так, чтобы каждой единице матрицы l соответствовал нуль в матрице k.
Замечание. «Ключ» можно поворачивать на угол, кратный 90°.

3.2 (10). Спираль. Дан массив $a(n, n)$, где n – нечетное число. Вывести элементы массива при обходе его по спирали, начиная с центра.

3.2 (11). Лабиринт. Лабиринт задан массивом $a(n, n)$, в котором элемент $a(k, m) = 0$, если клетка (k, m) «проходима», и $a(k, m) = 1$, если клетка «непроходима» (задается случайным образом). Начальное положение путника указывается в проходимой клетке (i, j). Путник может перемещаться по проходимым клеткам, имеющим общую сторону. Выяснить, может ли путник выйти из лабиринта (т.е. попасть в граничную клетку). Если да – распечатать путь.

3.2 (12). Латинский квадрат. Латинским квадратом порядка n называется квадратная таблица размером $n \times n$, каждая строка и каждый столбец которой содержит все числа от 1 до n. Для заданного n построить латинский квадрат.

3.2 (13). Дан целочисленный массив $a(n,m)$. Упорядочить по неубыванию элементы на главной диагонали и диагоналях, параллельных ей. Перестановка элементов допускается только в пределах соответствующей диагонали.

3.2 (14). Дан двумерный массив из 0 и 1. Удалить из массива столбцы и строки, содержащие одну 1. Если после исключения появляются новые строки и столбцы, удовлетворяющие этому условию, продолжить процесс исключения.

3.2 (15). Тени. В трехмерном массиве $k(l, m, n)$, состоящем из нулей и единиц, хранится сеточное изображение некоторого трехмерного тела. Получить в двумерных массивах три проекции (тени) этого тела.

3.2 (16). Крестики-нолики. Двумерный массив размером $m \times n$ есть результат игры в крестики-нолики на этом поле. Проверить, не закончена ли игра выигрышем «крестиков»? Считается, что «крестики» выиграли, если на поле найдется по горизонтали, вертикали или диагонали цепочка, состоящая подряд из 5 крестиков.

3.2 (17). Черный квадрат. В матрице $a(m, n)$, состоящей из нулей и единиц, найти квадрат заданного размера (квадратную подматрицу), состоящий целиком из нулей.

3.2 (18). 8 ферзей. Можно ли разместить на пустой шахматной доске 8 ферзей так, чтобы ни один из них не «нападал» на другого?

3.2 (19). Пирог. Имеется прямоугольный пирог, разрезанный на $m \times n$ частей (клеток), причем, левая нижняя клетка пирога отравлена. Игроки по очереди выбирают какую-нибудь клетку пирога и «съедают» ее

вместе со всеми клетками, расположенными правее и выше выбранной. Проигрывает тот, кто съедает отравленную клетку.

3.2 (20). Просветы. Куб состоит из n^3 прозрачных и непрозрачных элементарных кубиков. Имеется ли хотя бы один просвет по каждому из трех измерений? Если это так, вывести координаты каждого просвета.
Рекомендация. Для хранения куба выделить трехмерный массив с базовым типом минимально возможного размера.

3.2 (21). Обратная матрица. Найти матрицу, обратную заданной $a(n, n)$, методом Гаусса (в любой модификации).

3.2 (22). Волга-матушка. Гидрологами произведена серия замеров по прямой от берега до берега реки перпендикулярно фарватеру. Получены данные: s_i – расстояние от левого берега, м; h_i – глубина реки, м; v_i – скорость течения, м/с, $i = 1, 2,..., n$. Каков расход воды в сечении, т.е. сколько кубометров воды протекает через сечение реки в секунду?

3.2 (23). Соседи. Заполнить матрицу заданного размера $m(k, l)$ числами 1, 2, 3, 4 так, чтобы по горизонтали, вертикали и диагонали не было одинаковых рядом стоящих чисел.

3.2 (24). Улитка. Матрицу $a(m, n)$ заполнить натуральными числами от 1 до $m \times n$ по спирали, начинающейся в левом верхнем углу и закрученной по часовой стрелке.

3.2 (25). Непрозрачный куб. Куб состоит из n^3 прозрачных и непрозрачных элементарных кубиков. Построить полностью непрозрачный куб, используя ровно n^2 непрозрачных элементарных кубиков.
Рекомендация. Для хранения куба выделить трехмерный массив с базовым типом минимально возможного размера.

3.2 (26). Динамический двумерный массив $a(n,n)$ заполнить случайными числами. Удалить строку и столбец, содержащие максимальный элемент массив. Зеркально отобразить элементы массива относительно горизонтальной оси симметрии, а затем относительно вертикальной оси симметрии. Вывести на экран исходный и результирующий массивы.

3.2 (27). Массив $a(n, m)$ состоит из нулей и единиц. Требуется удалить из него совпадающие строки, а оставшиеся упорядочить по возрастанию двоичных чисел, формируемых из данных соответствующей строки.

3.2 (28). Элементы массива $a(n,m)$ ниже главной диагонали равны нулю. Выше и на главной диагонали нулевых элементов нет. Строки и столбцы случайным образом перемешаны. Перестановкой строк и столбцов восстановить исходный массив.

3.2 (29). Массив $a(n, n)$ разбивается на 4 четверти, ограниченные главной и побочной диагоналями: верхнюю, нижнюю, левую, правую. Поменять местами элементы верхней и нижней четвертей, правой и левой четвертей. Найти суммы элементов каждой четверти, без учета элементов, расположенных на диагоналях.

3.2 (30). Система уравнений. Решить систему уравнений $ax = b$ с квадратной матрицей a методом Гаусса. Предусмотреть возможность отсутствия и неединственности решения.

4. Строки

символьные массивы, строковые указатели, обработка текстовых файлов.

Контрольные вопросы

1. Что такое строка? Что является признаком конца строки в С? Какую строку называют нулевой?
2. Что произойдет с оставшимися байтами, если строка не заполняет полностью выделенную для нее память? Что произойдет, если выделенной памяти недостаточно?
3. Сколько элементов массива необходимо, чтобы запомнить строку “Minsk”? Сколько байт в памяти займет такой массив?
4. В чем заключается различие между массивами символов и указателями типа **char ***?
5. Как инициализировать массив символов **alpha** строкой “**ABCD**” двумя способами?
6. Что представляют собой следующие управляющие последовательности?
   1) `\n`
   2) `\\`
   3) `\"`
   4) `\b`
   5) `\a`
7. Какие действия выполняют функции **gets()**, **puts()**, **atoi()**, **atol()**, **atof()**? В каких заголовочных файлах они объявлены?
8. Какие действия выполняют функции **strcpy()**, **strncpy()**, **strcat()**, **strncat()**, **strcmp()**, **strncmp()**, **strchr()**, **strlen()**, **strdup()**, **strlwr()**, **strupr()**, **strstr()**, **strtok()**? В каком заголовочном файле объявлены эти функции?
9. Объясните, в каких случаях лучше при работе со строками использовать функцию **printf()**, а в каких **puts()**? **scanf()** и **gets()**? **strcpy()** и **strncpy()**?
10. Учитывает ли функция **strlen()** завершающий нулевой байт при подсчете символов в строке? Какое значение возвращает функция **strcmp()** в случае равенства и неравенства строк?
11. Сделано объявление: **char str[10] = "Hello";** вернут ли функции **strlen(str)** и **sizeof(str)** одно и то же значение?
12. Какие ошибки содержат следующие фрагменты программы?
    1)
    ```
    void main(void)
    {
      char city[] = {'B', 'r', 'e', 's', 't'};
    }
    ```
    2)
    ```
    char count[] = {'12345'};
    puts(count);
    ```

3)
```
char s[10];
strncpy(s, "hello", 5);
printf("%s\n", s);
```

4)
```
printf("%s", 'A');
```

5)
```
char s[12];
strcpy(s, "Welcome Home");
```

6)
```
if (strcmp(string1, string2))
  printf("The strings are equal\n");
```

7)
```
char s[] = "Вот массив символов";
for ( ; *s != '\0'; s++)
  printf("%c", *s);
```

13. Сравните эффективность методов прямого поиска подстроки в строке (тексте) и метода Боуера–Мура.

Примеры решения задач

Пример 4.1. Распечатать введенную строку, удалив '*****' и удвоив '**A**'.

ex41.c

```
#include <stdio.h>

void main(void)
{
  int i = 0;
  char str[80], ch;

  puts("\nВведите строку символов: ");
  while ((ch = getchar()) != '\n') { /* ввод символов */
    if (ch == '*') continue;
    if (ch == 'A' || ch =='a') str[i++] = ch;
    str[i++] = ch;
  }
  str[i] = '\0'; /* устанавливаем признак конца строки */
  puts(str);
}
```

В данном варианте программы в цикле ввода символов одновременно формируется выходной массив **str**. Можно вообще обойтись без массива, если в цикле **while** сразу осуществлять вывод нужных символов:

```
while ((ch = getchar()) != '\n') {
  if (ch == '*') continue;
  if (ch == 'A' || ch =='a') putchar(ch);
  putchar(ch);
}
```

Если пользователь введет строку более 80 символов, программа будет работать неправильно. В следующем примере показано, как этого избежать.

Пример 4.2а. Среди введенных слов распечатать сначала те, которые начинаются и оканчиваются одной и той же буквой, а затем – все остальные (слова выводить по одному на строке).

ex42a.c

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

#define LONG_STR 80 /* длина вводимой строки */
#define MAX_WORD 40 /* максимальное количество слов в строке */

void main(void)
{
  int i, k;
  char *word[MAX_WORD], *p, separator[] = " ,.?!;:",
    str[LONG_STR];

  printf("Введите строку слов длиной не более %d"
     "символов:\n", LONG_STR-1);
  gets(str);
  str[LONG_STR - 1] = '\0';       /* на случай недопустимо */
                                  /* длинной строки */
  k = 0;
  puts("\n\t\tВот слова в нужном порядке: \n");
  p = strtok(str, separator);
  while (p) {
    if (*p == *(p + strlen(p)-1)) /* если первая и последняя */
                                  /* буквы слова совпадают */
      puts(p);                    /* вывести его на экран */
    else word[k++] = p;           /* иначе запомнить в массиве*/
    p = strtok(NULL, separator);  /* p указывает на */
  }                               /* следующее слово */

  for (i = 0; i < k; i++)         /* вывод i-х элементов (слов)*/
    puts(word[i]);                /* двумерного массива word */
}
```

Чтобы корректно «разбить» строку на слова, используется функция **strtok()**. Даная функция воспринимает строку **str** как последовательность слов, разделенных символами, входящими в строку **separator**.

Функция возвращает указатель на первый символ **str**, не совпадающий с символами строки **separator** (другими словами, указатель на первое слово в **str**). Вызывая функцию **strtok()** повторно с параметром **NULL** в качестве первого аргумента, мы получим указатель на второе слово строки **str** и т.д. пока не дойдем до конца строки, после чего функция вернет значение **NULL**.

Следующая программа представляет собой модификацию этой задачи при работе с текстовыми файлами.

Пример 4.2b. В файле содержится список слов. Распечатать те из них, которые начинаются и оканчиваются одной и той же буквой.

ex42b.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define N 128                /* количество символов в строке */

void main(void)
{
  char buffer[N];
  FILE *fp;                  /* указатель на файл */

  fp = fopen("ex42b.txt", "r"); /* открыть файл ex42b.txt */
  if (!fp) exit(1);          /* если файл не найден – выход */
  /* считываем данные из файла в buffer построчно */
  while (fgets(buffer, N, fp) != NULL)
    if (buffer[0] == buffer[strlen(buffer)-2]) /* если первая */
                             /* буква слова совпала с последней*/
      printf(buffer);        /* печатаем слово */
  fclose(fp);                /* закрыть файл */
}
```

Функция **fopen()** открывает файл ex42b.txt в режиме чтения ("**r**"). Если открытие произошло успешно, функция возвращает ненулевое значение. В следующей строке мы проверяем это условие, чтобы в случае неудачи не продолжать выполнение программы.

Функция **fgets()** считывает данные из файла построчно. Т.к. в файле у нас хранится список слов, то после каждого вызова функции в строку **buffer** записывается очередное слово. После того, как будет считана последняя строка, функция возвращает **NULL**.

В операторе **if** мы проверяем совпадение первой и последней буквы слова. В квадратных скобках используется выражение **strlen(buffer)-2**, т.к. строки, копируемые в **buffer**, в конце содержат символы '**\n'** и '**\0'**.

Задачи

Требования к программам

1. Код программы должен быть логично разбит на функции.
2. Работа программы должна точно удовлетворять условиям задачи.
3. Программа не должна содержать предупреждений компилятора.
4. Программа должна быть хорошо протестирована.
5. Программа должна иметь дружественный интерфейс.
6. Код программы должен быть выдержан в хорошем стиле.

Указание к задачам 4.1 и 4.2

– Для решения задач не использовать стандартные строковые функции.

Задача 4.1

4.1 (1). С клавиатуры вводится многострочный текст. Отсортировать по длине и вывести на печать все слова текста.

4.1 (2). По правилам машинописи после запятой, двоеточия и точки с запятой в тексте всегда ставится пробел. После восклицательного, вопросительного знака и точки должен стоять пробел, а затем текст начинается с заглавной буквы. Составить программу исправления такого типа ошибок в тексте.

4.1 (3). С клавиатуры вводится многострочный текст. Оставить в тексте только первые вхождения каждой буквы, остальные – удалить. Вывести слова полученного текста, которые начинаются и заканчиваются одной и той же буквой.

4.1 (4). Дано *n* матриц. В каждой строке каждой матрицы записано одно слово. Если в слове нет повторяющихся символов – преобразовать все буквы слова в прописные, в противном случае – в строчные буквы. Полученные матрицы вывести.

4.1 (5). Разнобуквица. С клавиатуры вводится многострочный текст. Распечатать самое длинное слово, в котором все буквы разные.

4.1 (6). Из введенного пользователем многострочного текста распечатать только те слова, в которых буквы упорядочены по алфавиту.

4.1 (7). Для двух строк символов найти самую длинную общую подстроку. Пробелы и знаки препинания игнорировать, строчные и прописные буквы считать неразличимыми. Например, строки: “Дай вилку! Бок севрюжий кончается” и “Чемпионский кубок достался не нам” содержат общую подстроку “кубок”.

4.1 (8). Перевернутые слова. С клавиатуры вводится многострочный текст. Распечатать, начиная с последнего, «перевернутые» слова из текста.

4.1 (9). Строка содержит арифметическое выражение, состоящее из целых чисел и знаков операций: +, –, *, / (без скобок). Проверить корректность выражения (в смысле последовательности чисел и

знаков операций). В случае корректности вычислить значение выражения с учетом приоритетов операций.

4.1 (10). Морзянка. Вводимый с клавиатуры текст перевести в последовательность точек и тире с помощью азбуки Морзе.

Справка. Азбука Морзе:

А,A	.-	Б,B	-...	В,W	.--	Г,G	--.	Д,D	-..
Е,Ё,E	.	Ж,V	...-	З,Z	--..	И,I	..	Й,J	.---
К,K	-.-	Л,L	.-..	М,M	--	Н,N	-.	О,O	---
П,P	.--.	Р,R	.-.	С,S	...	Т,T	-	У,U	..-
Ф,F	..-.	Х,H	....	Ц,C	-.-.	Ч	---.	Ш	----
Щ,Q	--.-	Ъ	.--.-.	Ы,Y	-.--	Ь,X	-..-	Э	...-...
Ю	..--	Я	.-.-	1	.----	2	..---	3	...--
4	....-	5	.....	6	-....	7	--...	8	---..
9	----.	0	-----	.	.-.-.-	,	--..--	:	---...
?	..--..	‘,’	.----.	-	-....-	/	-..-.	;	-.-.-

4.1 (11). Отцентрировать (т.е. обеспечить осевую симметрию текста на экране добавлением пробелов слева) вводимый с клавиатуры текст. Первый символ помещается в 40-ю позицию; второй – в 41-ю; появление третьего и каждого последующего нечетного символа вызывает удаление одного пробела слева. Так продолжается до конца строки; ввод следующих строк – аналогично.

4.1 (12). Частотный словарь. Составить частотный словарь вводимого текста. Распечатать слова по алфавиту. Справа от каждого слова вывести частоту, с которой оно встретилось.

4.1 (13). Ввести с клавиатуры натуральное число n <= 1000 и символ, указывающий падеж. Вывести на экран введенное число русскими словами в заданном падеже.

4.1 (14). С клавиатуры вводится многострочный текст. Подсчитать частоту вхождения гласных латинских букв. Построить диаграмму.

4.1 (15). С клавиатуры вводится многострочный текст. Распечатать слова из текста, преобразовав их следующим образом:

- перенести последнюю букву в начало слова;
- оставить в слове только первые вхождения каждой буквы.

4.1 (16). Многострочный текст, введенный с клавиатуры, преобразовать следующим образом: каждая цифра заменяется на заключенную в круглые скобки последовательность знаков ‘+’ (если цифра четная)

или ‘–’ (если цифра нечетная), длина которой равна числу, представленному цифрой.

4.1 (17). Распечатать введенную строку, исключив из нее те символы, которые находятся между скобками ‘(’ ‘)’. Сами скобки не удалять. Если хотя бы одной скобки нет – сообщить об этом.

4.1 (18). Разработать рекурсивную функцию *StoI(n, str)*, которая преобразует строку десятичных цифр в целое число.

4.1 (19). **Цепочка слов.** Пусть слово – это последовательность от 1 до 8 символов, не включающая пробелов. Вводится n слов s_1, ..., s_n. Можно ли их упорядочить так, чтобы получилась «цепочка», в которой первая буква каждого слова s_i совпадает с последней буквой предыдущего слова, а последняя буква последнего слова совпадает с первой буквой первого слова? В цепочку должны входить все n слов без повторений. Если такое упорядочение возможно, то вывести цепочку слов.

4.1 (20). Определить, является ли периодической последовательностью заданная строка, т.е. имеет ли она вид $d\ d\ ...\ d$, где d – некоторая подпоследовательность символов.

4.1 (21). Разработать функцию *ItoB(n, s, b)*, которая переводит целое число n в строку s, представляющую число в системе счисления с основанием b.

4.1 (22). Часто встречающиеся ошибки начинающих наборщиков – дважды записанное слово и слово, записанное со строчной буквы после точки. Обнаружить и исправить такие ошибки во введенном многострочном тексте.

4.1 (23). Дано n матриц. Элементы матрицы – строки, в каждой строке записано одно слово. Удалить из матриц все палиндромы (слова, которые слева и справа читаются одинаково). Полученные матрицы вывести.

4.1 (24). Многострочный текст, введенный с клавиатуры, выровнять по ширине, вставляя дополнительные пробелы между словами (равномерно по всей строке).

4.1 (25). Во введенном многострочном тексте подсчитать количество слов и предложений. Вывести самое длинное слово и самую длинную строку.

4.1 (26). Даны две символьные строки, состоящие только из цифр (длина каждой – более 10 символов). Считая, что в этих строках находятся длинные целые числа, сформировать третью строку – сумму этих чисел.

4.1 (27). Поздравления. По заданному в массиве списку фамилий напечатать каждому упомянутому в списке поздравление к определенному празднику. Чтобы избежать шаблона, перечень желаемых благ выбирать как случайное подмножество из заготовленного списка (например, здоровья, счастья, продвижения по службе, долголетия и т.д.). Для универсальности программы сделать переменным и название праздника.

4.1 (28). С клавиатуры вводится многострочный текст. В каждом предложении поменять местами самое длинное и самое короткое слово. Подсчитать количество символов в самом длинном и самом коротком слове текста.

4.1 (29). Дано *n* матриц. В каждой строке каждой матрицы записано одно слово. Подсчитать сумму кодов символов каждого слова и, если сумма оказалась четной, развернуть зеркально это слово в строке. Полученные матрицы вывести.

4.1 (30). С клавиатуры вводится многострочный текст. Составить программу, которая проверяет правильность написания сочетаний "жи", "ши"; "ча", "ща"; "чу", "щу" и исправляет ошибки.

Задача 4.2

4.2 (1). В текстовом файле находится произвольный текст. Найти и распечатать пары слов (анаграммы), при прочтении которых в обратном направлении образуется другое слово пары, например: (ПОЛК КЛОП), (БАР РАБ).

4.2 (2). Текст в файле содержит многократно вложенные круглые скобки. Если баланс скобок соблюден, исправить текст, оставив скобки первого уровня круглыми, второго – заменить на квадратные, третьего и последующих – на фигурные.

4.2 (3). Текст в файле содержит положительные и отрицательные целые числа. Найти среднее арифметическое положительных чисел в файле.

4.2 (4). Длина строк текстового файла не превышает 80 символов. Разработать программу центрирования строк: короткие строки дополняются пробелами и размещаются по центру.

4.2 (5). Один из способов идентификации автора литературного произведения – подсчет частоты вхождения отдельных слов. В заданном текстовом файле найти 20 наиболее часто встречающихся слов с указанием частоты использования каждого из них.

4.2 (6). В текстовом файле хранится текст произвольной длины. Циклически сдвинуть каждую строку текста на *n* символов вправо, где *n* – порядковый номер строки в файле.
Пример: циклический сдвиг строки “abcdefg” на 3 символа дает результат: “efgabcd”.

4.2 (7). Текстовый файл содержит текст произвольной длины. Построить гистограмму распределения длин слов, которые он содержит.

4.2 (8). В текстовом файле находится произвольный текст. Разработать программу проверки правильности расстановки скобок (круглых, квадратных, фигурных). Критерий проверки: если встречается одна из закрывающих скобок, то последняя открывающая должна быть такого же типа; количество скобок каждого типа должно совпадать. Между скобками допустима запись любых символов.

4.2 (9). В файле-словаре найти и распечатать группы слов, записанных одними и теми же буквами (без учета регистра) и отличающиеся только их порядком, например, (КОМАР, КОРМА).

4.2 (10). В одном файле дан исходный текст программы на языке С, в другом – словарь ключевых слов этого языка. Преобразовать текст, записав ключевые слова прописными буквами, а остальные – строчными.

4.2 (11). Текстовый файл содержит текст произвольной длины. Построить вертикальную гистограмму числа вхождений в текст каждой строчной латинской буквы.

4.2 (12). Палиндромы. Текстовый файл содержит произвольный текст на русском языке. Найти в нем слова-палиндромы, одинаково читающиеся как слева направо, так и справа налево, например,

Анна, шалаш и т.д. Найти также и фразы-палиндромы, например: “А роза упала на лапу Азора” (пробелы, знаки препинания и регистр букв – игнорировать).

4.2 (13). Текст в файле содержит числа, которые записаны в восьмеричной системе счисления. Преобразовать эти числа в десятичную систему счисления и записать отредактированный текст обратно в файл.

4.2 (14). Текстовый файл содержит список ФИО студентов вашей группы. Упорядочить список по алфавиту. Проверить (и исправить, если нужно) написание собственных имен с прописных букв.

4.2 (15). В заданном текстовом файле подсчитать частоту использования каждого слова из словаря (другого текстового файла).

4.2 (16). В файле-словаре найти и распечатать слова, которые могут быть полностью составлены из других слов словаря с помощью конкатенации, например: “БАЛКОН” = “БАЛ” + “КОН”; “БАРСУК” = “БАР” + “СУК”.

4.2 (17). Латинизатор кириллицы. При интернет-общении с русской диаспорой в других странах часто возникает проблемы отсутствия кириллицы у зарубежных респондентов, а также слабое знание иностранных языков у соотечественников. Один из выходов – набор русских слов похожими по начертанию буквами латинского алфавита. Среди прописных русских букв таких насчитывается одиннадцать: A, B, E, K, M, H, O, P, C, T, X. Распечатать из заданного файла те слова, которые без искажения могут быть написаны латинскими буквами, например:

```
ВОВКА РОЕТ РОВ
СВЕТА СЕЕТ ОВЕС.
```

4.2 (18). В текстовом файле представлены названия лекарств, срок их годности, количество стандартов и стоимость одного стандарта (через пробелы), например:

```
Аспирин 15.04.99 127 1320
Пиковит 27.01.10 23 5340
```

Вывести на экран данные о лекарствах с истекшим сроком годности на сегодняшний день и подсчитать общую стоимость таких медикаментов.

4.2 (19). В файле задан произвольный текст. Напечатать в алфавитном порядке все слова, которые входят в этот текст по одному разу.

4.2 (20). Проанализировать текст в заданном файле. Вычислить среднее число букв в слове и слов в предложении. Вывести самое длинное слово и предложение на экран.

4.2 (21). Текстовый файл не содержит собственных имен и сокращений, набран с использованием прописных и строчных русских букв. Проверить, чтобы все предложения (и только они) начинались с прописной буквы; между словами был ровно один пробел. При необходимости откорректировать текст.

4.2 (22). По правилам пунктуации пробел может стоять после, а не перед каждым из следующих знаков: **. , ; : ! ?)] } ...**; перед, а не после знаков: **([{**. Заданный текстовый файл проверить на соблюдение этих правил и при необходимости исправить.

4.2 (23). Выяснить, верно ли, что в текстовом файле, состоящем только из цифр, букв и пробелов, сумма числовых значений цифр равна количеству слов.

4.2 (24). Пользователь вводит с клавиатуры слово. Найти в файле-словаре все слова, в которых использованы только буквы введенного слова.

4.2 (25). В файле дан исходный текст программы на языке С. Уровнем комментированности текста будем считать отношение объема комментариев к объему всего текста (в байтах). Определить уровень комментированности данного текста.

4.2 (26). Поле чудес. В слове угаданы некоторые буквы. Пользователь вводит «шаблон» слова, заменяя неизвестные буквы знаком подчеркивания. Компьютер из файла-словаря выбирает все слова, удовлетворяющие этому шаблону.

4.2 (27). Вычислить значения арифметических выражений, записанных в файл. Имя переменной – одна буква. Значения переменных в виде A = 3,1415, B = –37,2 и т.д. записаны во второй файл. Вывести на печать сами выражения и их вычисленные значения.

4.2 (28). Даны два текстовых файла. Первый файл содержит произвольный текст, второй – пары слов, каждая пара располагается в отдельной строке. Каждое первое слово в паре считается заменяемым, а второе – заменяющим. Найти в первом файле все заменяемые слова и заменить их на заменяющие.

4.2 (29). Текстовый файл состоит из 40-байтовых строк. Последние 8 байтов строки содержат номер строки (в случайном порядке). Упорядочить строки в файле по номерам.

4.2 (30). Произвольный текст в файле содержит даты. Каждая дата – это число, месяц и год (например, 03.05.1949). Вывести на печать наименьшую (самую раннюю) и наибольшую дату.

5. Структуры

структуры, стеки, очереди, списки, деревья, многоразрядные числа.

Контрольные вопросы

1. Что такое структура? Для чего в программировании используются структуры?
2. Являются ли следующие утверждения верными?
   1) Структуры могут содержать только один тип данных.
   2) Наличие псевдонима, определенного с помощью оператора **typedef**, является для структур обязательным.
   3) Члены одной структуры должны иметь уникальные имена.
   4) Структуры могут передаваться функциям только по значению.
   5) Динамические структуры стек и очередь являются частным видом списков.
3. Как можно проинициализировать структуру? Можно ли присваивать одной структуре другую? Можно ли их сравнивать?
4. Как можно обратиться к конкретному члену структуры? Как это сделать с помощью указателя на структуру?
5. Чем массив структур отличается от структуры массивов? Приведите примеры использования этих конструкций.
6. В чем заключаются преимущества и недостатки динамических структур данных по сравнению с обычными?
7. Что такое самоссылочные структуры? В чем заключается их основное преимущество? Какие наиболее распространенные самоссылочные структуры вы знаете?
8. Как называется вид списка, в котором элементы могут вставляться и удаляться только из конца? Вид списка, в котором элементы вставляются в конец, а удаляются из начала?
9. В чем заключается разница между однонаправленным и двунаправленным списком, между линейным и кольцевым списком?
10. Как наиболее быстро развернуть направление кольцевого списка?
11. Что такое дерево? Чем оно отличается от списков?
12. Что такое бинарное дерево поиска? В чем заключается его основное преимущество?
13. Какое максимальное число узлов может содержать бинарное дерево, имеющее три уровня?
14. Объявлена следующая структура:

```
struct list { int value; list *next; };
```

Что делают следующие функции?

```
int F1(list *p)
{
  int n;
  for (n = 0; p != NULL; p = p->next, n++) ;
  return n;
}

int F2(list *p, int n)
{
  for (; n != 0 && p != NULL; n--, p = p->next) ;
  return p->value;
}
```

15. Какие ошибки содержатся в следующих объявлениях структур?

1)
```
structure {
  char itable;
  int num[20]
  char *togs
}
```

2)
```
struct person {
  char lastName[15];
  char firstName[15];
  int age;
}
```

3)
```
struct key {
  int number;
};
key d;
```

4)
```
struct key {
  int number;
} *ptr;
ptr->number = 128;
```

5)
```
struct card {
  char *face;
  char *suit;
} *ptr;
printf("%s", *ptr->face);
printf("%s", *ptr.suit);
```

Примеры решения задач

Пример 5.1. Разработать функцию, суммирующую два многоразрядных числа. Для хранения многоразрядного числа использовать динамический двунаправленный список.

ex51.c

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

```
#include <alloc.h>
#include <string.h>
/* Структура, описывающая элемент двунаправленного списка */
typedef struct item {
  int digit;
  struct item *next;
  struct item *prev;
} Item;
/* Структура, описывающая многоразрядное число */
typedef struct mnumber {
  Item *head;
  Item *tail;
  int n;
} MNumber;

MNumber CreateMNumber(char *initStr);
void AddDigit(MNumber *number, int digit);
void PrintMNumber(MNumber number);
MNumber SumMNumber(MNumber n1, MNumber n2);

void main(void)
{
  MNumber a = CreateMNumber("123456789123456789");
  MNumber b = CreateMNumber("987654321987654321");
  MNumber c = SumMNumber(a, b);
  PrintMNumber(a);
  PrintMNumber(b);
  PrintMNumber(c);
}
/* Создает многоразрядное число из цифр строки */
MNumber CreateMNumber(char initStr[])
{
  MNumber number = {NULL, NULL, 0};
  int n;

  for (n = strlen(initStr)-1; n >= 0; n--)
    AddDigit(&number, initStr[n]-'0');
  return number;
}
/* Добавляет цифру в многоразрядное число */
void AddDigit(MNumber *number, int digit)
{
    Item *p = (Item *)malloc(sizeof(Item));
    p->digit = digit;
    p->next = p->prev = NULL;
    if (number->head == NULL)
      number->head = number->tail = p;
    else {
      number->tail->next = p;
      p->prev = number->tail;
      number->tail = p;
    }
```

```
    number->n++;
}
/* Возвращает сумму двух многоразрядных чисел */
MNumber SumMNumber(MNumber n1, MNumber n2)
{
  MNumber sum = CreateMNumber("");
  Item *p1 = n1.head, *p2 = n2.head;
  int digit, pos = 0, s1, s2;

  while (p1 || p2) {
    if (p1) { s1 = p1->digit; p1 = p1->next; }
    else s1 = 0;
    if (p2) { s2 = p2->digit; p2 = p2->next; }
    else s2 = 0;
    digit = (s1 + s2 + pos) % 10;
    pos = (s1 + s2 + pos) / 10;
    AddDigit(&sum, digit);
  }
  if (pos) AddDigit(&sum, pos);
  return sum;
}
/* Выводит многоразрядное число на экран */
void PrintMNumber(MNumber number)
{
   Item *p = number.tail;

   printf("\nNumber: ");
   while (p) {
     printf("%d", p->digit);
     p = p->prev;
   }
}
```

Структура **MNumber**, описывающая многоразрядное число, содержит указатели на начало и конец динамического двунаправленного списка, в котором будут храниться цифры числа, и количество цифр в нем – **n**.

В функции **main()** создаются два многоразрядных числа – **a** и **b**, их сумма присваивается числу **c**. После этого все три числа выводятся на экран.

Функция **CreateMNumber()** принимает инициализирующую строку, в которой записаны цифры числа. Вначале в функции создается «пустое» многоразрядное число, а затем последовательно, элемент за элементом, значения из строки копируются в двунаправленный список. Обратите внимание, что строку мы копируем, начиная с конца: для арифметических расчетов удобнее, чтобы список «рос» от младших разрядов к старшим.

В функции **SumMNumber()** происходит суммирование многоразрядных чисел. Цикл **while (p1 || p2)** выполняется, пока не закончатся разряды в обоих слагаемых. На каждой итерации вычисляется очередная цифра суммирующего числа **digit** и «перенос» на следующий разряд **pos**.

Если после выполнения цикла появился перенос в самом старшем разряде, то к сумме добавляется еще один (старший) разряд.

Функция **PrintMNumber()** выводит число в обратном порядке, от «хвоста» к «голове» списка, т.к. мы добавляли цифры, начиная с конца числа.

В виде полезного упражнения модифицируйте программу, чтобы пользователь мог сам инициализировать переменные **a** и **b**; разработайте функцию **DeleteMNumber()**, удаляющую передаваемое ей многоразрядное число.

Пример 5.2. В текстовом файле содержится набор чисел. С помощью бинарного дерева поиска вывести числа на печать по возрастанию.

ex52.c

```
#include <stdio.h>
#include <stdlib.h>
#include <alloc.h>
/* Структура, описывающая узел дерева */
typedef struct item {
  int data;
  struct item *left;
  struct item *right;
} Item;

void AddNode(int data, Item **node);
void LeftOrder(Item *node);

void main(void)
{
  char buffer[128];
  int i;
  Item *root = NULL;  /* корень дерева */
  FILE *fp = fopen("ex52.txt", "r");

  if (!fp) exit(1); /* выйти, если файл не удалось открыть */
  /* Считываем значения из файла построчно */
  while (fgets(buffer, 128, fp) != NULL)
    AddNode(atoi(buffer), &root); /* и добавляем в дерево */
  LeftOrder(root);
  fclose(fp);
}
/* Добавить узел в бинарное дерево поиска */
void AddNode(int data, Item **node)
{
  if (*node == NULL) {
    *node = (Item *)calloc(1, sizeof(Item));
    (*node)->data = data;
    (*node)->left = (*node)->right = NULL;
  } else {
    if (data < (*node)->data)
      AddNode(data, &(*node)->left);
```

```
    else if (data > (*node)->data)
      AddNode(data, &(*node)->right);
    else
      puts("There is such element in the tree");
  }
}
/* Обход дерева слева (вывод по возрастанию) */
void LeftOrder(Item *node)
{
  if (node->left)
    LeftOrder(node->left);
  printf("%d ", node->data);
  if (node->right)
    LeftOrder(node->right);
}
```

В начале программы объявляется структура для работы с бинарным деревом. В функции **main()** после открытия текстового файла значения в цикле считываются в строку **buffer**, пока не будет достигнут конец файла. С помощью стандартной функции **atoi()** цифры, записанные в строке, преобразуются в целые значения, которые и добавляются в дерево.

Т.к. функция **AddNode()** строит бинарное дерево поиска, обход слева этого дерева выводит значения по возрастанию (обход справа вывел бы их по убыванию).

Задачи

Требования к программам

1. Код программы должен быть логично разбит на функции.
2. Работа программы должна точно удовлетворять условиям задачи.
3. Программа не должна содержать предупреждений компилятора.
4. Программа должна быть хорошо протестирована.
5. Программа должна иметь дружественный интерфейс.
6. Код программы должен быть выдержан в хорошем стиле.

Задача 5.1

5.1 (1). Дан двунаправленный закольцованный список, каждый элемент которого содержит указатель на стек. Элементы стеков содержат символы. Отсортировать по алфавиту элементы каждого стека и вывести их на печать. Выполнить слияние стеков в один упорядоченный по алфавиту стек и распечатать его.

5.1 (2). Разработать функцию разложения многоразрядного числа на простые множители. Для хранения многоразрядного числа использовать динамический двунаправленный список.

5.1 (3). С помощью двунаправленного списка определить, является ли строка палиндромом (т.е. одинаково читается справа налево и слева направо). Пробелы и знаки пунктуации – игнорировать.

5.1 (4). Разработать функцию, переводящую многоразрядное десятичное число в восьмеричную систему. Для хранения многоразрядного числа использовать динамический двунаправленный список.

5.1 (5). Дан стек, каждый элемент которого содержит указатель на очередь. Элементы очередей хранят целые числа. Вывести на печать очередь, которая содержит наибольшее число и очередь, сумма элементов которой максимальна. Выполнить слияние очередей в один стек и распечатать его.

5.1 (6). **Автосервис.** Автосервис предоставляет n различных услуг, в каждую из которых – очередь автомобилей. Новые автомобили заезжают в автосервис через случайные интервалы времени; время обслуживания в каждом из n сервисов – постоянная величина a_i ($i <= n$). Выбрать маршрут, при котором за заданное время t автомобиль сможет пройти максимальное число сервисов.

5.1 (7). Разработать функцию вычисления 3^n, при $n > 100$. Для хранения многоразрядного числа использовать динамический двунаправленный список.

5.1 (8). Даны две упорядоченные очереди. Разработать функцию, которая объединяет их в один упорядоченный двунаправленный список.

5.1 (9). Напишите программу, которая использует стек для того, чтобы определить, является ли строка палиндромом (т.е. одинаково читается справа налево и слева направо). Программа должна игнорировать пробелы и знаки пунктуации.

5.1 (10). Проверить, является ли число $2^{19936}\times(2^{19937}-1)$ совершенным, т.е. равным сумме своих делителей, кроме себя. Для хранения многоразрядного числа использовать динамический двунаправленный список.

5.1 (11). Задана очередь, указывающая порядок обработки одного из n стеков. Элементы очереди содержат номер обрабатываемого стека i

(i <= n); признак выполняемого действия: ‘A’ – элемент из очереди добавляется в *i*-й стек, ‘D’ – из *i*-го стека удаляется элемент; данные (целые число *d*). Разработать функцию обработки стеков в соответствии с заданной очередью.

5.1 (12). Разработать функцию вычисления *n*!, при *n* > 100. Для хранения многоразрядного числа использовать динамический двунаправленный список.

5.1 (13). Разработать функцию, которая инвертирует заданную очередь, т.е. первый элемент становится последним, второй – предпоследним и т.д.

5.1 (14). Определить наибольший общий делитель двух многоразрядных чисел. Для хранения многоразрядного числа использовать динамический двунаправленный список.

5.1 (15). Определить количество делителей многоразрядного числа. Для хранения многоразрядного числа использовать динамический двунаправленный список.

5.1 (16). Дан стек, каждый элемент которого содержит указатель на двунаправленный список. В каждом элементе списков хранится слово. С помощью рекурсивной функции вывести каждое слово в обратном порядке. На основе списков составить один упорядоченный по алфавиту стек и распечатать его.

5.1 (17). Даны два упорядоченных стека, в которых могут быть одинаковые элементы. Разработать функцию создания одного общего упорядоченного стека, исключив повторяющиеся элементы.

5.1 (18). **Супермаркет**. Напишите программу, которая моделирует очередь в кассу супермаркета. Покупатели подходят к кассе через случайные интервалы времени, на обслуживание каждого покупателя также тратится случайное время. Через заданное время *T* программа должна завершиться. Вывести на экран:

- максимальное количество покупателей в очереди за время *T*;
- максимальное время, которое покупателю пришлось ждать в очереди.

Замечание. Для решения задачи использовать динамическую структуру очередь. Работу программы представить в графическом виде на экране.

5.1 (19). Разработать функцию, переводящую многоразрядное десятичное число в двоичную систему. Для хранения многоразрядного числа использовать динамический двунаправленный список.

5.1 (20). Даны два упорядоченных списка, в которых могут быть одинаковые элементы. Разработать функцию создания одного общего упорядоченного списка, исключив повторяющиеся элементы.

5.1 (21). Создать тип данных *Многоразрядное число*. Разработать следующие функции:

- **Equal()** – сравнение двух многоразрядных чисел (возвращает 0, если числа равны, 1 – если первое число больше, -1 – если второе число больше);
- **LongModShort()** – возвращает остаток от деления многоразрядного числа на короткое число типа **int**;
- **LongDivShort()** – возвращает результат целочисленного деления многоразрядного числа на короткое число типа **int**;
- **LongMulShort()** – возвращает результат умножения многоразрядного числа на короткое число типа **int**.

Для хранения многоразрядного числа использовать динамический двунаправленный список.

5.1 (22). **Призывник**. Призывник должен пройти медкомиссию из 5 кабинетов, в каждый из которых стоит очередь. Призывники заходят в поликлинику через случайные интервалы времени и становятся в очередь в тот кабинет, в котором меньше очередь. Время, которое каждый из них проводит в кабинете, – также случайно. Через какое время призывник пройдет медкомиссию?
Замечание. Для решения задачи использовать динамическую структуру очередь. Работу программы проиллюстрировать на экране.

5.1 (23). Даны две упорядоченные очереди, в которых могут быть одинаковые элементы. Разработать функцию создания одной общей упорядоченной очереди, исключив повторяющиеся элементы.

5.1 (24). Создать тип данных *Многоразрядное число*. Разработать следующие функции:

- **LongSumLong()** – сложение двух многоразрядных чисел;
- **LongSubLong()** – вычитание двух многоразрядных чисел;
- **LongMulLong()** – умножение двух многоразрядных чисел;
- **LongDivLong()** – деление двух многоразрядных чисел.

Для хранения многоразрядного числа использовать динамический двунаправленный список.

5.1 (25). Разработать функцию, которая инвертирует заданный стек, т.е. первый элемент становится последним, второй – предпоследним и т.д.

5.1 (26). Определить наименьшее общее кратное двух многоразрядных чисел. Для хранения многоразрядного числа использовать динамический двунаправленный список.

5.1 (27). Задан двунаправленный список, указывающий порядок обработки одной из *n* очередей. Элементы списка содержат номер обрабатываемой очереди *i (i <= n)*; признак выполняемого действия: 'A' – элемент из списка добавляется в *i*-ю очередь, 'D' – из *i*-й очереди удаляется элемент; данные (символ). Разработать функцию обработки очередей в соответствии с «программой» двунаправленного списка.

5.1 (28). Разработать функцию, которая инвертирует заданный список, т.е. первый элемент становится последним, второй – предпоследним и т.д.

5.1 (29). Определить, являются ли два многоразрядных числа взаимно простыми. Для хранения многоразрядного числа использовать динамический двунаправленный список.

5.1 (30). Напишите функцию копирования одного стека в другой.

Задача 5.2

5.2 (1). Даны два бинарных дерева поиска. Разработать функцию, которая создает одно общее бинарное дерево поиска, исключая повторяющиеся элементы.

5.2 (2). Дан двунаправленный список, элементы которого содержат номер обрабатываемого дерева i ($i \le n$) и копируемые в дерево данные (целое число *d*). Сформировать на основе стека *n* бинарных деревьев поиска. Вывести на печать наибольшие элементы в каждом дереве.

5.2 (3). Дан стек, элементы которого содержат номер обрабатываемого дерева i ($i \le n$) и копируемые в дерево данные (целое число *d*).

Сформировать на основе стека *n* бинарных деревьев поиска. Подсчитать и вывести на печать сумму значений в каждом дереве.

5.2 (4). В текстовом файле записаны целые числа. Построить бинарное дерево поиска, в узлах которого хранятся числа из файла. Разработать функцию, определяющую число узлов дерева на каждом уровне.

5.2 (5). Построить бинарное дерево поиска из букв строки, вводимой пользователем. Разработать функцию подсчета числа листьев дерева.

5.2 (6). Дано кольцо, элементы которого содержат целые числа. Сформировать на основе кольца бинарное дерево поиска. Вывести самую длинную ветвь в получившемся дереве.

5.2 (7). Пользователь вводит произвольную строку. Построить на основе входящих в строку чисел бинарное дерево поиска и вывести числа упорядоченными по возрастанию и по убыванию.

5.2 (8). Пользователь вводит с клавиатуры целые числа (ввод прекращается, когда будет введен ‘0’). Построить бинарное дерево поиска, каждый узел которого содержит целое число. Повторяющиеся значения в дерево не добавлять. Разработать функцию проверки сбалансированности дерева (для каждого узла количество вершин в левом и правом поддеревьях различается не более чем на 1).

5.2 (9). В текстовом файле содержатся *n* целых чисел в двоичной системе счисления (*m* бит каждое). Построить бинарное дерево, в котором числам соответствуют листья дерева, а путь по дереву определяется двоичным представлением числа (‘1’ – переход к правому потомку, ‘0’ – переход к левому потомку).

5.2 (10). Пользователь вводит слова с клавиатуры (ввод прекращается, когда будет введено слово “end”). Построить бинарное дерево поиска, каждый узел которого содержит слово, и вывести по алфавиту те слова, которые являются палиндромами (т.е. читаются справа налево и слева направо одинаково).

5.2 (11). Разработать функции сравнения и копирования бинарного дерева.

5.2 (12). Задана очередь, указывающая порядок обработки одного из *n* бинарных деревьев поиска. Элементы очереди содержат: номер

обрабатываемого дерева i ($i \le n$); признак выполняемого действия: ‘A’ – элемент из очереди добавляется в i-е дерево, ‘D’ – из i-го дерева удаляется заданный элемент; данные (целое число d). Обработать деревья в соответствии с заданной очередью.

5.2 (13). Разработать рекурсивную функцию **OutputTree()**, отображающую на экране бинарное дерево. Функция должна выводить дерево ряд за рядом; корень дерева должен располагаться в левой части экрана, а листья – справа. Каждый ряд выводится вертикально, например:

```
                99
          97
                92
     83
                72
          71
                69
49
                44
          40
                32
     28
                19
          18
                11
```

5.2 (14). В текстовом файле содержится произвольный текст. Построить на его основе бинарное дерево поиска, каждый узел которого содержит слово. Вывести из дерева на печать слова, начинающиеся и заканчивающиеся на одну и ту же букву, упорядочив их по алфавиту.

5.2 (15). В текстовом файле записаны целые числа. Построить сбалансированное бинарное дерево, в котором количество левых и правых потомков у каждой вершины отличается не более чем на единицу.

5.2 (16). Построить бинарное дерево поиска из букв строки, вводимой пользователем. Разработать функцию, подсчитывающую, сколько уровней содержит дерево.

5.2 (17). Разработать функцию, которая на основе произвольного бинарного дерева строит бинарное дерево поиска. Повторяющиеся элементы удалить.

5.2 (18). Дана очередь, элементы которой содержат целые числа. Сформировать на основе очереди бинарное дерево поиска. Разработать функцию «обрезки» дерева, которая оставляет на каждом узле только один лист.

5.2 (19). В текстовом файле записаны целые числа. Построить бинарное дерево поиска, в узлах которого хранятся числа из файла. Разработать функцию, определяющую количество нечетных элементов.

5.2 (20). Пользователь вводит с клавиатуры целые числа (ввод прекращается, когда будет введен ‘0’). Построить бинарное дерево поиска, повторяющиеся значения в дерево не добавлять. Разработать функцию, возвращающую сумму значений узлов самой длинной ветви дерева.

5.2 (21). На квадратном поле m на n посеяно $m \times n$ саженцев бинарных деревьев поиска. В текстовом файле содержится набор натуральных чисел. Каждое число из файла записать в создаваемый узел случайно выбранного дерева и таким образом вырастить сад. Распечатать по возрастанию и убыванию самое «плодовитое» дерево (сумма значений узлов которого максимальна).

5.2 (22). Дан двунаправленный список, элементы которого содержат целые числа. Сформировать на основе списка бинарное дерево поиска. Создать два стека: в один поместить все четные числа из дерева, в другой – нечетные. Распечатать стеки.

5.2 (23). В текстовом файле содержится произвольный текст. Построить на его основе бинарное дерево поиска, каждый узел которого содержит слово. Вывести на печать те слова, в которых встречаются три и более гласных букв латинского алфавита.

5.2 (24). Построить бинарное дерево поиска из букв строки, вводимой пользователем. Разработать функцию, которая для заданной буквы определяет число узлов в правом и левом поддеревьях.

5.2 (25). Разработать функцию, которая принимает в качестве аргумента троичное дерево и определяет, сколько уровней оно содержит.

5.2 (26). Дан связный список, элементы которого содержат целые числа. Сформировать на основе списка бинарное дерево поиска. Разработать функцию удаления из дерева всех узлов, содержащих отрицательные значения.

5.2 (27). В текстовом файле содержится произвольный текст. С помощью бинарного дерева определить частоту вхождения каждого слова в текст. Вывести слова по убыванию частоты вхождения.
Примечание. Каждый элемент дерева должен хранить слово. Кроме этого, необходимо вести счетчик числа вхождений этого слова в текст.

5.2 (28). Построить бинарное дерево поиска из букв строки, вводимой пользователем. Разработать функцию, вычисляющую, каких букв больше – гласных или согласных.

5.2 (29). В узлах бинарного дерева расположены строки. Создать стек и поместить в него все строки, содержащиеся в листьях дерева и имеющие четную сумму кодов символов.

5.2 (30). Напишите программу, которая вводит строку текста, разделяет предложение на отдельные слова, вставляет слова в бинарное дерево поиска, а затем распечатывает их, обходя дерево слева и справа.

6. Файлы и базы данных

текстовые и бинарные файлы, разработка простейших СУБД

Контрольные вопросы

1. Что такое файл? В каких целях используются файлы?
2. Верны ли следующие утверждения?
   1) Функцию **fscanf()** можно использовать для чтения данных с клавиатуры.
   2) С помощью **fprintf()** можно вывести информацию на экран.
   3) Программа должна явно вызывать функцию **fclose()** для того, чтобы закрыть файл.
   4) В файле произвольного доступа нет необходимости просматривать все записи для того, чтобы найти определенную запись.
   5) Записи в файлах произвольного доступа могут иметь разную длину.
   6) Функция **fseek()** может производить поиск только от начала файла.
3. В чем различие между текстовыми и бинарными файлами? В чем заключаются преимущества тех и других?
4. Какие вы знаете режимы открытия текстовых и бинарных файлов?
5. Режимы “**a+**”, “**r+**” и “**w+**” открывают файл и для записи и для чтения. Какой из них является наиболее подходящим для изменения уже существующей информации в файле?
6. Объясните, что такое посимвольный и построчный режимы обработки файлов, последовательный и произвольный доступ. К каким типам файлов относятся эти понятия?
7. Какие действия выполняют функции **fopen()**, **fclose()**, **fgetc()**, **fgets()**, **fputc()**, **fputs()**, **fprintf()**, **fscanf()**?
8. Что делают следующие функции?

```
void F1(FILE *ifp, FILE *ofp)
{
  int c;
  while ((c = getc(ifp)) != EOF)
    putc(c, ofp);
}

void F2(char *str, FILE *iop)
{
    int c;
    while (c = *str++)
      putc(c, iop);
}
```

9. Как можно очистить уже существующий файл?
10. Как можно организовать быстрый постраничный просмотр текстового файла, не храня его полностью в памяти, но и не загружая каждый раз заново?
11. Как можно узнать позицию указателя в файле без использования функции **ftell()**?
12. Как можно сохранить в файле дерево?
13. Что неправильно в этой программе?

```
void main(void)
{
  int k, *fp;
  fp = fopen("counting");
  for (k = 0; k < 30; k++)
    fputs(fp, "One, two, three...");
  fclose("counting");
}
```

Примеры разработки технического задания

Пример 6.1. Автовокзал

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Проект: «**Касса автовокзала**»
Разработал: **Ярмолик Виталий**

1. Vision Statement

Программа предназначена для просмотра расписания движения пригородных и междугородних автобусов. В расписании находится информация о времени отправления, промежуточных пунктах, количестве свободных мест на данном рейсе. В программе реализована возможность заказа билетов, просмотра и редактирования посадочной ведомости, возможность выбора ближайшего рейса до заданного пункта, возврат билетов. Предусмотрена возможность полного редактирования расписания, как с добавлением рейсов, так и с удалением их. К программе прилагается справочное руководство.

Предполагаемым пользователем программы является англоязычная женщина от 25 до 50 лет со знанием текстового редактора Notepad или пакета Microsoft Office (в частности, Microsoft Word).

Данная программа может применяться на автовокзале в качестве основного инструмента для управления расписанием и уведомления пассажиров об изменениях в расписании.

2. Структуры данных

```
typedef enum
{
        Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday
} Days;
typedef struct time {                    // структура для описания времени отправл.
```

```
        int  hours;                 // час отправления
        int  minutes;               // минуты отправления
        Days  day;                  // день отправления
} Time;

typedef struct trips {              // структура, описывающая характ. рейса
        int      number;            // номер рейса
        char   terminate[30];       // конечный пункт
        char   *interm[15];         // массив промежуточных пунктов
        Time  departure;            // время отправления
        int      spaces;            // количество свободных мест
        struct trips  *next;        // указатель на следующий эл. списка
        struct trips  *prev;        // указатель на предыдущий эл. списка
} Trips;
```

3. Архитектура системы

Программа состоит из четырех модулей:

Основной модуль (basicunit.h)

В этом модуле содержится описание структур trips, time. Также в данном модуле содержится описание и реализация следующих функций:

– чтение и запись расписания в файл

– вывод расписания на экран

– добавление и удаление рейса из расписания

– поиск ближайшего рейса до заданного пункта

Модуль для проведения операций с билетами (tickets.h)

В этом модуле содержится описание и реализация следующих функций:

– покупка билетов

– просмотр посадочной ведомости

– редактирование посадочной ведомости

– возврат билетов.

Справочный модуль (help.h)

В этом модуле содержится описание и реализация функций для работы со справочной системой.

Интерфейсный модуль(interface.h)

В данном модуле находятся функции для работы с меню: отображение меню, контроль ввода пользователя, отображение заголовков уже пройденных вершин меню.

Пользователю доступны все функции, предназначенные для работы с билетами и расписанием.

Меню программы имеет следующий вид:

A Ticket Window For Bus Terminal

1. View timetable
2. Edit timetable
3. View seat list
4. Edit seat list

5. Find appropriate race

6. Buy ticket

7. Return ticket

8. Help

9. Exit

Описание пунктов меню:

1. View timetable

После выбора данного пункта меню на экран выводится расписание в виде таблицы со следующими названиями столбцов: Number of trip (номер рейса), Terminate (конечный пункт), Intermediate points (промежуточные пункты), Departure (время отправления), Spaces (количество свободных мест).

2. Edit timetable

После выбора данного пункта меню на экране появляется подменю со следующим содержанием:

2.1 Add new trip

2.2 Delete trip

При добавлении нового рейса пользователь должен указать его характеристики (номер, конечный пункт, промежуточные пункты, время отправления, количество мест). При удалении рейса необходимо указать номер удаляемого рейса.

3. View seat list

Выбор данного пункта меню позволят просмотреть посадочную ведомость.

4. Edit seat list

После выбора данного пункта меню на экране появляется подменю со следующим содержанием:

4.1 Add new seat

4.2 Delete seat

Данные пункты меню позволяют редактировать посадочную ведомость (изменять количество свободных мест)

5. Find appropriate race

После выбора данного пункта меню происходит поиск ближайшего рейса до конечного пункта заданного пользователем.

6. Buy ticket

После выбора данного пункта меню происходит оформление билета и его покупка. Возможна покупка билета, как на ближайший рейс, так и на последующие рейсы (указывается дата поездки).

7. Return ticket

Данный пункт меню позволяет осуществить возврат уже купленного билета. Возможен возврат билета на определенное время, как на ближайший рейс, так и на последующие (указывается дата поездки).

8. Help

Выбор данного пункта меню позволяет воспользоваться справкой к программе.

9. Exit

После выбора данного пункта меню происходит запрос на подтверждение выхода из программы и выход из нее при положительном ответе пользователя.

4. Системные требования

Системные требования минимальны: процессор – 100 MHz, оперативная память – 16 MB, 1 MB свободного пространства на диске. Операционные системы MS-DOS, Windows.

Пример 6.2. Супермаркет

ТЕХНИЧЕСКОЕ ЗАДАНИЕ ПО ПРОЕКТУ СУПЕРМАРКЕТ v1.0

Колесов М.Ю.
гр. 552001

Обзор программы

Программа Супермаркет предназначена для автоматизации управления покупками товаров в магазинах. Она является удобным инструментом в осуществлении контроля над приобретаемым товаром, его количеством, методом оплаты товара и наличием скидки при его покупке. Программа позволяет контролировать распределение скидок с помощью системы дисконтных карт. Полезной возможностью программы является способность хранить всю информацию о покупках в базе данных, из которой в любой момент можно взять информацию о совершенной ранее сделке.

Программа рассчитана на пользователя, владеющего русским или английским языком. Программа имеет дружественный интерфейс и может быть использована пользователем, имеющим минимальный уровень компьютерной грамотности.

Структура данных

В программе использованы следующие структуры данных:

1) для хранения информации о товарах:

Типы:

```
//штрих-код фиксированной длинны
typedef char[10] tBarcode;

//цена может быть только положительной и целой (для BYB)
typedef unsigned long tPrice;

//наименование товара
typedef char[30] tName;

//количество данного товара (на складе либо купленного)
typedef unsigned long tGoodsNumber;
```

Структура:

```
//структура для хранения информации об одиночном товаре
typedef struct {
      tBarcode Barcode;
      tPrice Price;
      tName Name;
      tGoodsNumber GoodsNumber;
      //для организации связного списка
      tGoods *Next, *Prev;
```

```
} tGoods;
```

2) для хранения информации о дисконтных картах:

Типы:

```
//процент скидки
typedef unsigned char tDiscount;

//номер дисконтной карты фиксированной длинны
typedef char[15] tDiscoutCardCode;
```

Структура:

```
//структура для хранения информации о дисконтной карте
typedef struct {
      tDiscount Discount;
      tDiscoutCardCode Code;
      //для организации связного списка
      tDiscountCard *Next, *Prev;
} tDiscountCard;
```

3) для хранения информации о кредитных картах:

Типы:

```
//сумма на счете
typedef unsigned char tCardSumm;

//номер кредитной карты фиксированной длинны
typedef char[20] tCreditCardCode;

//пин-код кредитки
typedef char[4] tPinCode;
```

Структура:

```
//структура для хранения информации о дисконтной карте
typedef struct {
      tCardSumm CardSumm;
      tCreditCardCode Code;
      tPinCode PinCode
      //для организации связного списка
      tCreditCard *Next, *Prev;
} tCreditCard;
```

4) для хранения истории о покупках:

Типы:

```
//представление даты покупки
typedef struct {
      char Year;
      char Month;
      char Day;
      char Hour;
      char Min;
      char Sec;
} tDealDate;

//количество различных типов купленного товара
typedef unsigned int tDifferentGoodsNumber;

//сумма покупки
typedef unsigned long tSumm;
```

```
typedef enum {FALSE=0, TRUE=1} tBoolean;
```

Структура:

```
//структура, содержащая информацию о покупке
typedef struct {
      tDealDate DealDate;
      tDifferentGoodsNumber DifferentGoodsNumber;
      tSumm Summ;
      //использовалась ли дисконтная карта
      tBoolean IfUsedDiscount;
      //если использовалась дисконтная карта – здесь ее скидка
      tDiscount Discount;
      //если использовалась дисконтная карта – здесь ее код
      tDiscoutCardCode UsedDiscountCardCode;
      //использовалась ли кредитная карта
      tBoolean IfUsedCreditCard;
     //если использовалась кредитная карта – здесь ее код
      tCreditCardCode UsedCreditCardCode;
      //указатель на список купленных товаров
      tGoods *ListOfBuyedGoods;
      //для организации связного списка
      tDeal *Next, *Prev;
} tDeal;
```

Архитектура системы

Работа с программой организована в виде навигации по различным пунктам меню.

Карта меню:

1) Режим обслуживания покупателей[1]
2) Работа с историей покупок
   1) Отображение всей истории покупок
   2) Поиск в истории покупок по разным критериям
   3) Редактирование истории покупок (ограниченный доступ)
   4) Сортировка истории по разным критериям
   5) Возврат в предыдущее меню
   6) Выход из программы
3) Работа со списком товаров на продажу
   1) Отображение всего списка товаров
   2) Поиск товара по разным критериям
   3) Редактирование списка товаров (ограниченный доступ)
   4) Сортировка списка товаров по разным критериям
   5) Возврат в предыдущее меню
   6) Выход из программы
4) Работа с информацией о дисконтных картах
   1) Отображение всего списка дисконтных карт
   2) Поиск дисконтной карты по разным критериям
   3) Редактирование списка дисконтных карт (ограниченный доступ)
   4) Сортировка списка дисконтных карт по разным критериям

[1] Представляет собой режим ввода покупок очередного покупателя сразу же после предыдущего, не возвращаясь в данное меню. Позволяет быстро и эффективно осуществлять регистрацию покупок, суммирование цены, вычисление цены со скидкой (при наличии у покупателя дисконтной карты) и регистрацию метода оплаты (наличный расчет, расчет кредитной картой). Возможны отмена введенных данных на любом уровне.

    5) Возврат в предыдущее меню
    6) Выход из программы
5) Смена языка интерфейса
6) Помощь
7) Информация о разработчике
8) Выход из программы

Доступ к пунктам меню осуществляется нажатием на клавиатуре клавиш цифр, соответствующих позиции в меню. При выборе пункта меню, программа ведет пользователя далее с помощью интуитивно понятного интерфейса, который предусматривает обработку корректных и некорректных действий пользователя.

На каждой странице, после выбора пункта меню, внизу экрана находится список «горячих» клавиш, осуществляющих быстрый вызов часто используемых действий на данной странице. В нижней части окна находится краткое описание функциональности данной страницы.

В программе имеется список общих «горячих» клавиш, с помощью которых возможен доступ к общим свойствам и действиям различных пунктов меню. Список «горячих» клавиш доступен из пункта меню Помощь.

Системные требования

Компьютер под управлением операционной системы Windows (любой версии) или DOS. Аппаратная часть достаточна, если работают названные операционные системы.

Задачи

Выполнение данной лабораторной работы состоит из двух этапов. На первом этапе необходимо разработать текстовый документ *Техническое задание* (ТЗ). Второй этап – разработка на основе утвержденного технического задания *программного продукта*.

Общие требования

1. Работа программы должна точно удовлетворять условиям технического задания.
2. Программа должна иметь удобный и дружественный пользователю интерфейс.
3. Программа не должна содержать предупреждений (Warnings) компилятора.
4. Программа должна быть хорошо протестирована: при любом наборе корректных входных данных она должна работать правильно.
5. Код программы должен быть выдержан в хорошем стиле. Необходимы комментарии в ключевых местах.

Специальные требования

1. Программа должна представлять собой проект, состоящий из нескольких модулей.
2. Каждый модуль должен быть логично разбит на функции.

Требования к качеству

Независимо от предметной области, выделяются требования, на которых защищается квалификация разработчика.

1. **Устойчивость программы.** Программа не должна терять работоспособности ни при каких, даже некорректных, действиях пользователя. Любые действия, грозящие потерей информации, выполняются только после повторного подтверждения. Вводимая информация там, где возможно, подвергается логическому контролю.
2. **Полнота информации.** Вся информация поддается просмотру и редактированию. Удаляемую информацию полезно переносить в архивные базы для последующего просмотра и восстановления. Записи, содержащие много полей, можно просматривать как в табличном, так и в постраничном виде (в виде карточек).
3. **Поиск данных**. Программа должна позволять проводить поиск или выборку информации по любому полю в базе данных.
4. **Порядок движения.** Движение по меню (вход в подменю) сопровождается заголовками всех пройденных вершин; возврат возможен только на предыдущий уровень с сохранением введенной информации.
5. **Терминологическая среда и интерфейс.** В сообщениях используется язык и термины, понятные пользователю, а не разработчику («запись», «индексация», «can't open file» и т.д. – не допустимы). Язык диалога – с соблюдением норм вежливости, цветовая гамма – по общепринятым рекомендациям.
6. **Использование клавиатуры.** На любом этапе нажатие любой клавиши (особенно функциональных) должно игнорироваться или вызывать предусмотренные действия, описанные в средствах помощи. Привязка действий к клавишам – общепринятая: F1 – помощь, Enter – согласие, завершение ввода; Esc – отказ, возврат к родительскому меню; Tab – переход к следующему полю, окну и т.д.; Shift-Tab – возврат к предыдущем полю и т.д.
7. **Средства помощи.** В любой точке алгоритма в строке подсказки высвечиваются все активные в данный момент горячие клавиши; в любой момент при нажатии клавиши F1 выдается контекстно-зависимый (зависящий от ситуации) текст помощи.
8. **Рекламная заставка.** При запуске программы появляется рекламная заставка, отражающая суть и возможности программного средства, а также сведения об авторе. Такую заставку можно разработать средствами C либо сделать в других редакторах (PowerPoint, Word, Flash и т.д.) и запускать отдельно.

Указания к задаче 6.1

Предлагаемая формулировка представляет собой лишь поверхностные наброски постановки и не претендует на завершенность. Она примерно соответствует той информации, которую может дать в начале работы над проектом заказчик. Конкретизация и уточнение возлагаются на разработчика. Необходимо изучить предметную область, выяснить, кто является потенциальным пользователем системы, какая информация и для чего используется. После этого на основе первоначальной формулировки разрабатывается и согласовывается с заказчиком (преподавателем) техническое задание.

Четких стандартов на написание ТЗ нет. Каждая компания использует свой стандарт, а небольшие группы программистов делают ТЗ так, как им удобно, или ориентируются на требования заказчика. В нашем случае ТЗ должно представлять собой текстовый документ, который включает в себя:

1. **Заголовок.** В начале страницы указывается название документа: «Техническое задание», название проекта, ФИО разработчика.
2. **Описание.** В Microsoft этот раздел называется Vision statement – *видение программного продукта*. В первом абзаце записывается, *что* будет делать программа, каково ее *назначение*, какие *главные функции* она будет выполнять. В следующем абзаце описывается портрет предполагаемого пользователя системы (примерный возраст, пол, на каком языке говорит, уровень компьютерной грамотности и т.д.). Далее необходимый уточняющий текст располагается в произвольной форме.
3. **Архитектура.** Описание системы с точки зрения пользователя – какие интерфейсы и функции ему будут доступны. Необходимо продумать структуру меню и описать, что произойдет при выборе того или иного пункта.
4. **Структура данных.** Описываются создаваемые типы данных и используемые динамические структуры. Комментируется каждое поле.
5. **Системные требования**. Перечисляются программные и аппаратные средства, необходимые для запуска программы.

Указания к задаче 6.2

Общая последовательность работы с программой следующая: вначале открывается файл базы данных и информация считывается в динамический двунаправленный список. После этого пользователю предлагается выбор функций, утвержденных в ТЗ. Пользовательский интерфейс может быть реализован в виде текстового или графического меню. После завершения работы с программой информация из динамического списка записывается обратно в файл, вся динамическая память освобождается, а файл закрывается.

Задача 6.1

6.2 (1). **Книжный магазин**. Прием партии книг на склад; по каждому наименованию указывается: автор, название, реквизиты, количество экземпляров, цена приемки, отпускная цена. Оформление заказа: выбор и продажа нескольких наименований в заданном количестве экземпляров. Отчетность: количество имеющихся на складе книг; их общая стоимость; количество сделанных заказов на книги; количество проданных книг; общая прибыль магазина за период с момента внедрения программы. Списки имеющихся книг, история продаж; функция просмотра. Сохранение введенных данных на диск.

6.2 (2). **Общежитие БГУИР.** База данных блоков и комнат в общежитии. Заселение формируется по принципу факультетов: за каждым факультетом закреплены «свои» этажи. Операции заселения и выселения. Формирование списка жильцов (ФИО, факультет, курс, группа, нарушения). При наличии более трех нарушений правил проживания, программа выводит напоминание о необходимости выселения. Список служащих (заведующая, паспортистка, психолог, воспитатели, комендант, старосты этажей), содержит их контактную информацию. Поиск комнаты по фамилии, фамилий по комнате. Отчетность: количество жильцов, количество и номера свободных мест (для всего общежития и для каждого факультета в отдельности).

6.2 (3). **Абитуриент**. Программа для приемной комиссии БГУИР. Формирование списка факультетов и специальностей. Подача заявления на специальность (ФИО, оценки по трем предметам, средний балл аттестата, паспортные данные, место проживания и номер школы абитуриента). Абитуриент имеет право забрать заявление. Зачисление по специальностям согласно поданным заявлениям и выделенным бюджетным местам. Определение проходных и полупроходных баллов по специальностям и факультетам. Отчеты: рейтинг абитуриентов, список поступивших на данную специальность, рейтинг специальностей и факультетов.
Справочные сведения: http://abitur.bsuir.by

6.2 (4). **Администратор гостиницы.** Список всех номеров, которые подразделяются на классы (обычный класс, полулюкс, люкс, королевский номер), число мест. Списки занятых и свободных номеров. Поселение гостей: выбор подходящего номера (при наличии свободных мест), регистрация, оформление квитанции. Отъезд: выбор всех постояльцев, отъезжающих сегодня,

освобождение места или оформление задержки с выпиской дополнительной квитанции. Возможность досрочного отъезда с перерасчетом. При заселении создается список постояльцев: ФИО, паспортные данные, номер комнаты, дата приезда и выселения. Поиск постояльца по любому полю. Расчет стоимости проживания.

6.2 (5). **B-Project.** Программа B-Project представляет собой упрощенную версию продукта MS Project, позволяющего управлять проектами. Каждый проект состоит из задач, которые могут выполняться последовательно или параллельно. Задача имеет дату начала и конца. На проект выделяются люди, которые будут работать над ним, и ресурсы (если это необходимо). Программа рассчитывает сроки проекта, ресурсы, определяет критический путь и задачи, которые могут выполняться параллельно, запас времени на их выполнение.

6.2 (6). **cShop.by**. Интернет-магазин занимается реализацией компьютерной техники и комплектующих. В каждом разделе существуют подразделы и сами товары. Формирование списка товаров по запросу пользователя. Доступная информация о товаре: название, цена, наличие на складе, описание, гарантия, страна производства и т.д. После выбора продукта, он добавляется в корзину и вычисляется общая цена покупки. При подтверждении покупки, формируется список покупателей: ФИО, контакты, список покупок. Отчеты о продажах, общая сумма выручки, история покупок.

6.2 (7). **Социальная сеть.** Множество людей организовано по принципу сети: каждый человек имеет свой круг общения, у каждого из этого круга, в свою очередь, есть свой круг и т.д. Добавление нового члена (ФИО, контакты, город, образование, работа, интересы). Установление и удаление связей. Поиск нужных людей по интересам, местонахождению, образованию или работе. Возможность просмотра кругов общения, т.е. «путешествия» по социальной сети.
Подробная информация: www.moikrug.ru .

6.2 (8). **Супермаркет**. Программа для кассового аппарата. Имеется база уникальных штрих-кодов (представлены в виде числовых значений). С каждым штрих-кодом связано название товара и его цена. При вводе задается количество нужного товара (ничего не введено – 1 штука, введен 0 – отмена ввода). Вести историю всех покупок. После окончания ввода всех товаров одного покупателя программа спрашивает, есть ли дисконтная карта, и если есть, то требует ее серийный номер. Сверяет по базе дисконтных карт, берет

оттуда процент скидки, производит расчет и записывает эту информацию в историю покупки.

6.2 (9). **HotelTracer.** Программа работает с базами данных семи мировых туристических сайтов: www.HotelQuest.com, www.TravelWeb.com, www.opodo.com, www.hotels.com, www.orbitz.com, www.AllHotels.com, www.TravelWorm.com. Каждая база данных представляет собой двоичный файл, содержащий информацию о свободных номерах в отеле: город, название отеля, адрес отеля, класс номера (обычный класс, полулюкс, люкс, vip-номер), количество мест и стоимость номера. Информация в базах данных обновляется каждые 60 секунд. HotelTracer ищет наилучший отель, соответствующий запросу пользователя. Примеры запросов: “Венеция, престижный номер на двоих, цена не имеет значения”, “София, одиночный номер по минимальной цене”. Функции бронирования номера и снятия брони.

6.2 (10). **Приорбанк**. Сведения о вкладчиках банка: номер лицевого счета, категория вклада, паспортные данные, текущая сумма вклада (остаток). Операции приема и выдачи сумм по вкладу (разрешенные категорией вклада), расчет процентов на заданную дату. Переводы денежных средств между клиентами, подготовка квитанций. Отчеты: выписка счета (все операции, проведенные с данным счетом в текущем году); список всех клиентов банка и поиск по нему; сортировка по остаткам вкладов клиентов; сумма остатков.
Справка: информация по условиям вкладов: www.prior.by

6.2 (11). **Путеводитель.** Разработать программу для карманного электронного путеводителя. Международный поиск по городам (в каждом городе существует список мест отдыха (название, адрес, направление деятельности, время работы) – отели, кинотеатры, музеи, ночные клубы, театры и др.). Информация о памятниках и сооружениях, имеющих историческую ценность. Пользователь может выбрать город и получить подробную информацию о местах отдыха. Поиск оптимального отеля по запросу пользователя. Сформировать записную книжку тех мест, где он побывал, с возможностью добавлять комментарии.

6.2 (12). **Стипендия.** Разработать программу для расчетной группы БГУИР, начисляющей стипендию. Ввод, редактирование и удаление информации о студенте, его среднем балле. Расчет стипендии. Начисление именных стипендий при наличии ходатайства кафедры. Печать ведомости: список студентов, которым была начислена стипендия, общее количество выплаченных денег. Список

студентов, получающих повышенную стипендию. Социальная надбавка.
Справка: 5-6 – 78,862 руб., 6-8 – 94,634 руб., 8-9 – 110,405 руб., 9-10 – 126,179 руб. Стипендия Совета вуза – 135,000 руб., именная стипендия – 181,000 руб.

6.2 (13). **Ломбард**. База хранимых товаров и недвижимости: наименование товара, идентификатор клиента, оценочная стоимость, сумма, выданная под залог, дата сдачи, срок хранения. Список клиентов (ФИО, паспортные данные, дата регистрации, история обслуживания). Редактирование, удаление и просмотр введенной информации. Функции приема/возврата товара, расчет прибыли ломбарда, оценочная стоимость всех хранимых вещей.

6.2 (14). **Областная больница**. Больница состоит из нескольких отделений. В каждом отделении находится определенное количество палат. Программа формирует список отделений (название, месторасположение), список врачей (ФИО, должность, специальность), списки пациентов (ФИО, адрес, год рождения, диагноз, дата поступления, дата выписки номер палаты). Для каждого пациента назначается свой курс лечения, он вводится пользователем программы. Подготовка к печати истории болезни. Расчет нагрузка на врача и отделение. Если отделение переполнено – вывод возможной даты принятия нового больного.

6.2 (15). **Биржа труда.** База претендентов: ФИО, год рождения, контакты, пол, специальность, образование, семейное положение, город проживания, полная или частичная занятость, желаемый уровень з/п. База вакансий: название компании, город, должность, специальность, образование, пол, семейное положение, возраст, частичная или полная занятость, уровень оплаты. Поиск и регистрация вариантов с той и другой стороны; удаление в архив после трудоустройства, полное удаление при отказе от услуг. Перечень всех возможных трудоустройств по имеющейся БД.
Подробная информация: www.jobs.tut.by .

6.2 (16). **Рейтинг студентов**. Программа рассчитывает рейтинг студентов специальности «Информатика». Рейтинг учитывает все оценки, полученные студентами во время сессий – как положительные, так и отрицательные. Можно просмотреть рейтинг по курсам, по группам. Отображается текущий средний балл, рост (положительный или отрицательный) относительно последней сессии, изменение места в рейтинге.

6.2 (17). DeltaTest. Разработать локальную систему тестирования. Провести тестирование по языку программирования C.

6.2 (18). Банкомат. Имеется база пользователей. У каждого пользователя на счету *n*-ая сумма. Для входа в систему используется имя и пароль. Запись истории всех действий со счетом – как входящих средств, так и исходящих. Пользователи могут переводить средства между собой. При входе вывести экран приглашения для ввода имени и пароля. Регистрацию новых пользователей можно реализовать в окне приветствия. Изначально запустить в систему достаточную сумму.

6.2 (19). Outlook. Программа реализует ежедневник, содержащий контактный лист владельца (ФИО, контакты, место работы, группа (друзья, коллеги, родственники и т.д.), дата рождения) и данные о делах (звонки, встречи, задания, пометки). Каждое мероприятие хранит в себе дату, время, продолжительность, место проведения, ссылки на контакты, примечание. Программа автоматически генерирует напоминание о днях рождениях. Возможно заполнение ежедневника периодическими событиями на год. Анализ накладок. Возможность просмотра мероприятий на любую дату.

6.2 (20). Библиотека кафедры. Покупка новой книги: автор, название книги, стоимость, примечание; покупка новых экземпляров уже имеющейся книги. Добавление читателя: фамилия, имя, номер группы, телефон, электронная почта, примечание. Возможности редактирования и удаления введенной информации. Функции выдачи (указывается дата возврата) и возврата книг. Желтая и красная карточка читателям, которые не вернули книги вовремя. Отчетность: количество книг в библиотеке, их общая стоимость, количество книг на руках, количество читателей. Просмотр и поиск книг и читателей.

6.2 (21). mp3All.com. Интернет-магазин mp3-композиций. Программа хранит список музыкальных направлений и статьи о них, список исполнителей (одиночные исполнители и группы). Каждый исполнитель имеет свой список альбомов и композиций, а также тексты композиций и информацию о себе. Поиск композиции и исполнителя по всем полям. Помещение выбранных композиций в «корзину», функция оплаты. Отчеты: количество композиций, дисков и т.д., количество проданных композиций и выручка магазина.

6.2 (22). Квадратный метр. Программа для риэлтерской компании. База предложения недвижимости: район и адрес, этаж, площадь, тип планировки, цена. База спроса: требования покупателя к жилью (возможно несколько вариантов, допустимые диапазоны, например: “однокомнатная до 40.000, Малиновку и Шабаны не предлагать”), финансовые возможности, контакты. Регистрация собственников и покупателей, поиск по всем полям, подбор вариантов для той и другой стороны, автоматизированный поиск взаимоприемлемых вариантов. Удаление при состоявшейся сделке или отказе от услуг. Функция обмена жилья (при совпадающих условиях обмена).

6.2 (23). Source Manager. Программа представляет собой сборник исходных кодов на разных языках программирования и набор статей и комментариев к ним. Пользователь может добавлять и редактировать комментарии к каждому исходнику. Исходные коды и статьи можно только просматривать (они загружаются из файлов). У исходника и статьи есть данные: язык программирования, тематика, дата добавления. Программа позволяет выбрать исходник по тематике либо языку программирования и просмотреть код, сопутствующие статьи и комментарии.

6.2 (24). Santaren. Программный модуль управления туристическими программами. В каждую программу входит следующая информация: маршрут, дата, количество мест, общая стоимость тура. Программа состоит из набора обязательных и необязательных туристических услуг. Пользователь может самостоятельно сформировать свою программу либо выбрать полную программу. Возможно создание новых туристических программ и добавление услуг в существующие. Запись туристов на каждую программу, оформление туристических поездок по мере комплектования групп. Расчет доходов компании.

6.2 (25). Брачное агентство. База потенциальных женихов и невест: регистрационный номер, ФИО, пол, дата рождения, рост и вес, город, интересы, вредные привычки, информация о себе, требования к партнеру. Поиск подходящих вариантов по запросу, подготовка встреч (формирование приглашения для знакомства). Автоматизированный поиск взаимоприемлемых вариантов. Перенос в архив пар, решивших свои семейные проблемы, удаление клиентов, отказавшихся от услуг.

6.2 (26). Отдел кадров. Содержит полный список сотрудников организации: ФИО, дата рождения, паспортные данные, карточка соц. страхования, образование, должность, оклад, дата поступления на

работу, примечание (поощрения и взыскания). Программа осуществляет поиск сотрудника по любому полю. Функции увольнения и принятия на работу. Показывает срок работы сотрудника и подыскивает время отпуска (после полугода работы полагается 2 недели отпуска). Сообщения о приближающихся днях рождения сотрудников. Списки работников, которые находятся на работе и в отпуске. Сокращение штатов: выбор для увольнения лиц пенсионного и предпенсионного возраста, подготовка приказа.

6.2 (27). Интерпол. База Интерпола содержит информацию о международных преступлениях и преступниках. База преступлений (место преступления, участники, последствия, комментарии специалистов), база преступных и террористических организаций (название, род деятельности, рейтинг опасности), база преступников (ФИО, кличка, рост, вес, цвет волос, особые приметы, гражданство, знание языков, род преступления, причастность к террористическим организациям, ссылки на список преступлений, ссылки на список преступных организаций). Просмотр и поиск информации по всем полям. Установление связей между преступниками и между организациями. Анализ, кто мог совершить данное преступление. Выборка «завязавших» в архив; удаление – только после смерти.

6.2 (28). BelAvia. Компания работает по заданному годовому расписанию: номер рейса, тип самолета, маршрут, промежуточные пункты, время отправления, дни полета. Поиск по всем полям. Выбор ближайшего рейса до заданного пункта. Функции заказа билетов на рейс и возврата билетов. Количество свободных мест на каждом рейсе. Оформление посадочной ведомости, печать билетов.

6.2 (29). FruitImport. Белорусская компания импортирует фрукты из ряда европейских стран. Компания арендует торговые площадки, находящиеся на двух главных маршрутах: 1) в Варшаве, Праге, Вене, Милане, Марселе, Барселоне и 2) в Киеве, Кишиневе, Бухаресте, Софии, Афинах. FruitImport располагает 6 грузовиками, каждый из которых способен перевезти максимум 25 тонн груза. В каждой машине едет 2 водителя, таким образом, машина находится в пути круглосуточно. Средняя скорость – 60 км/ч, среднее время на пересечение границы – 4 часа. Стоимость бензина и зарплата водителям – стандартные. В Минске компания сразу реализует привезенные фрукты оптовику, ее прибыль составляет 5-15% от выручки. Каждый час, в течение суток, партнеры передают компании информацию о поставках фруктов (объем (тонн) и цена партии) на торговых площадках. Разработать программу

оптимального движения грузовиков FruitImport. Проиллюстрировать движение автомобилей и совершаемые деловые операции в режиме реального времени.

6.2 (30). **Атлант-М**. База новых и подержанных автомобилей: марка, год выпуска, объем двигателя и др. технические характеристики, состояние, цена. База покупателей: ФИО, контакты, желаемая марка, год выпуска, технические характеристики и состояние, финансовые возможности. Поиск по всем полям. Сортировка автомобилей по цене, по году выпуска. Подбор вариантов для покупателя, формирование заявки для поставщиков. Расчет прибыли компании.

Справочные сведения. Себестоимость доставки автомобиля из Европы составляет для компании в среднем €250. В конечную стоимость включается также цена растамаживания, зависящая от объема двигателя и «возраста» автомобиля:

- 3 года и менее – €0,6 за 1 см.куб.;
- от 3 до 10 лет включительно: с рабочим объемом цилиндров двигателя до 2500 см.куб. – €0,35 за 1 см.куб.; с рабочим объемом цилиндров двигателя 2500 см.куб. и более – €0,6 за 1 см.куб.;
- от 10 до 14 лет – €0,6 за 1 см.куб.;
- 14 и более лет – €2 за 1 см.куб.

Конечная цена устанавливается так, чтобы обеспечить компании 15% норму прибыли.

Задача 6.2

Разработать программное обеспечение по техническому заданию своего варианта.