где n — число вершин многоугольника, x, y — указатели на массивы координат вершин. Реализуйте функции, обеспечивающие работу с многоугольниками, как с объектами:

```
double  Perimeter(struct Polygon p);                         /* периметр */
double  Area(struct Polygon p);                              /* площадь */
struct Polygon * Clip(struct Polygon a, struct Polygon b);/* пересечение */
int     Equal(struct Polygon a, struct Polygon b);           /* равны ? */
int     Convex(struct Polygon a);                            /* выпуклый ? */
```

Считаем, что пересечение непересекающихся многоугольников пусто. Функция `Clip` создает новый многоугольник, т.е. выделяет для него память и заполняет требуемыми значениями.

1.220. Пусть в трехмерном пространстве определена плоскость — задан вектор нормали и трехмерные координаты одной точки. Реализуйте функцию, которая вычисляет координаты ортогональной проекции точки пространства на эту плоскость.

1.221. Пусть в трехмерном пространстве дана плоскость — два базисных вектора и трехмерные координаты одной точки. Реализуйте функцию, которая вычисляет координаты ортогональной проекции точки пространства на эту плоскость.

1.222. Пусть в трехмерном пространстве дана плоскость с локальной системой координат — два базисных вектора и трехмерные координаты точки на плоскости, являющейся началом координат. Реализуйте функцию, которая вычисляет локальные координаты ортогональной проекции точки пространства на эту плоскость.

1.9 Рекурсия.

1.223. Напишите рекурсивную и последовательную реализации вычисления функции $n!$, сравните их эффективность (см. 1.16, 1.30).

1.224. Напишите рекурсивную реализацию для вычисления дисперсии $D = \frac{1}{n}\sum\limits_{i=1}^{n}(x_i - M)^2$ элементов числовой последовательности. Сравните эффективность с 1.45).

1.225. Оцените размер программного стека, выделяемого в Вашей системе.
Решение. В программном стеке для каждой вызываемой функции хранятся все локальные переменные класса `auto`, возвращаемое значение и адрес оператора возврата. Поэтому каждый рекурсивный вызов функции

```
void  f(int n){
  char buf[1016];
  printf("n=%d\n",n);
  f(n+1);
  return ;
```

будет уменьшать размер стека на `(sizeof(buf)+sizeof(n)+sizeof(void*))` байт.

1.226. Золотым сечением Φ называется положительный корень уравнения $x^2 - x - 1 = 0$, т.е. число $\Phi = (1+\sqrt{5})/2$. Его можно определить рекурсивно $x = 1 + 1/x$, т.е. с помощью цепной дроби $\Phi = 1 + \frac{1}{1+\frac{1}{1+\dots}}$. Экспериментально найти количество слагаемых цепной дроби, обеспечивающих точность 10^{-10}.

Указание. Рекуррентная формула имеет вид: $x_{n+1} = \phi(x_n) = 1 + \frac{1}{x_n}$. Покажите, что процесс сходится к Φ при $n \to \infty$ для произвольного $x_0 > 0$. Для этого можно использовать т.н. паутинную диаграмму Ламерея — изображение графиков функций $\phi(x) = 1 + 1/x$ и $g(x) = x$ с отмеченной на них траекторией $x_0, \phi(x_0), \phi^2(x_0), \dots$, задаваемой точками x_0, $y_0 = \phi(x_0)$, $x_1 : g(x_1) = y_0$, $y_1 = \phi(x_1)$, $x_2 : g(x_2) = y_1$, ...

1.227. Напишите рекурсивную и циклическую реализации вычисления n–го числа Фибоначчи $x_n = x_{n-1} + x_{n-2}$, $x_1 = x_2 = 1$, см. задачу 1.30. Сравните время работы программ.

1.228. Напишите рекурсивные реализации приближенного вычисления числа e, используя следующие цепные дроби:

$$1)\, e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \dots; \quad 2)\, e = 2 + \frac{1}{1+\frac{1}{2+\frac{2}{3+\frac{3}{4+\dots}}}},$$

$$3)\, e = [2; 1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8, \dots], \text{ где } [a_0; a_1, a_2, a_3, \dots] = a_0 + \frac{1}{a_1+\frac{1}{a_2+\frac{1}{a_3+\dots}}}$$

Сравните скорости сходимости соответствующих итерационных процессов.

Идеи реализации. Вторую формулу можно вычислить рекурсивно, используя соотношение $e = 2+\frac{1}{h(1)}$, где $h(n) = n+n/h(n+1)$. Итерации следует прекратить, если на очередном шаге полученное приближение изменилось менее, чем $10^{-10}\%$.

1.229. Напишите рекурсивную реализацию приближенного вычисления числа e, используя следующее соотношение: $(e+1)/(e-1) = g(2)$, где $g(n) = n + 1/g(n+4)$. Сравните скорости сходимости данного метода и методов из задачи 1.228.

1.230. Напишите рекурсивную реализацию алгоритма сортировки простым слиянием, см. 1.154.

1.231. Для заданных положительных чисел $a < b$ найти и выписать все представления вида $b = a[\,][\,]\dots[\,]$, где каждую скобку можно заменить на одно из следующих выражений $\{[+2], [+3], [*5]\}$. При этом считаем, что полученной формуле все действия выполняются последовательно слева направо независимо от приоритета операций.

1.232. Для заданных положительных чисел $a < b$ и n найти и выписать все представления $b = a[\,][\,]\dots[\,]$ длины не более n, где вместо каждой из скобок можно подставить одно из следующих выражений $\{[+2], [-3], [*5]\}$. В формуле действия выполняются последовательно слева направо.

1.233. Пусть заданы целое число a, множество из двух чисел $B = \{b_1, b_2\}$, множество из двух бинарных арифметических действий $Q = \{-, /\}$. Требуется найти и распечатать всевозможные представления единицы в виде

$$1 = a * [\,] * [\,] * \ldots * [\,],$$

где вместо каждой из $*$ можно ставить произвольное действие из Q, а вместо $[\,]$ — произвольное число из B. В формуле действия выполняются последовательно слева направо.

1.234. Пусть заданы целое число a, множество $B = \{b_1, b_2, \ldots, b_n\}$ из n различных целых чисел, множество из четырех бинарных операций $Q = \{-, /, +, \cdot\}$ и целое число N_0. Требуется найти и распечатать, см. 1.235, всевозможные представления единицы в виде

$$1 = a * [\,] * [\,] * \ldots * [\,],$$

содержащие не более N_0 арифметических действий. В формуле действия выполняются последовательно слева направо.

1.235. Пусть задан двумерный целочисленный массив А размерности M на N, заполненный нулями и двойками. Двойки соответствуют стенкам, а нули — пустотам. С экрана вводятся координаты i, j некоторого элемента массива. Требуется смоделировать процесс заливки пустот из (i, j)-ой ячейки, т.е. необходимо заполнить единицами те элементы массива, до которых можно добраться, последовательно перемещаясь в соседние нулевые ячейки.

1.236. Пусть задан двумерный целочисленный массив А размерности M на N, заполненный нулями и двойками. Двойки соответствуют стенкам, а нули — проходам в лабиринте. С экрана вводятся координаты i, j. Требуется найти всевозможные пути, по которым можно добраться из (i, j)-ой ячейки до ячейки $(0, 0)$, не пересекая ячеек со значением два. Результат распечатать на экране в виде последовательности указаний выбора направления.

1.237. Требуется вывести в файл все k-элементные подмножества множества $\{0, \ldots, N-1\}$.

Идеи реализации. Создадим массив из N элементов-признаков, соответствующих элементам исходного подмножества и указывающих использован или нет данный элемент в подмножестве. Формирование подмножества выполняется рекурсивной процедурой, которая последовательно помечает один свободный элемент массива как использованный в подмножестве и обращается сама к себе. Глубину рекурсивных вызовов нужно ограничить значением k. Также см. 1.199.

1.238. Составьте программу, которая для введенного натурального числа выводит его значение “словами”. Например, для числа 427 выводит строку “четыреста двадцать семь”.

1.239. Пусть имеется некоторое количество однозначных чисел. Построим арифметическое выражение, используя все эти числа и объединяя их операциями сложения, умножения, вычитания, деления, возведения в степень.

Составьте программу, которая по заданному массиву однозначных чисел вычисляет значения всевозможных таких выражений и печатает их вместе с видом соответствующего выражения. При этом порядок чисел в выражении всегда остается одним и тем же, а действия выполняются слева направо без учета приоритета операций.

1.240. Решите задачу 1.239 со следующими усложнениями:

1) можно использовать скобки для группировки операций;

2) можно изменять порядок следования чисел в выражении;

3) можно “склеивать” несколько подряд идущих чисел для образования многозначного числа.

1.241. Пусть имеется файл, содержащий целочисленную последовательность неизвестной заранее длины. Требуется за один просмотр файла сохранить последовательность в памяти компьютера, а затем результат распечатать на экране.

Идеи реализации. Для хранения элементов будем использовать следующую структуру *рекурсивного типа*

```
struct List{
   int A;
   struct List *next;
};
```

сохраняя в переменной `A` текущей структуры очередной элемент последовательности, а в переменной `next` указатель на структуру для следующего элемента (`NULL` в последней структуре). В этом случае рекурсивная функция печати может иметь, например, следующий вид:

```
void prn(struct List *cur){
  static int num=0;
  num++;
  if(num==1){
    printf("\n<");
  }
  if(cur!=NULL){
   printf(" %d ",cur->A);
   cur=cur->next;
   prn(cur);
  }
  else{
   printf(">\n");
 }
return;
}
```

— А что такое рекурсия?

— Да посмотри в Википедии.

— Уже смотрел... Написано: См. Рекурсия.