1. **Ввод и вывод вещественных данных.** Наиболее приятный вид имеет числовая информация, организованная по табличному типу – в виде колонок фиксированной ширины, в которых одноименные числовые разряды располагаются друг под другом. При этом более рационально используется площадь экрана. Достигается это за счет управления форматами выводимых данных.

В программе описывается массив из 20 вещественных чисел. Осуществить запросколичества чисел и их ввод. Вывести их компактно (в одну или несколько строк) в предположении, что вводимые числа вещественные и имеют 2 значащие цифры в дробной части. Перед каждым числом разместить его порядковый номер и после номера «:». После нажатия любой клавиши вывести их столбиком, располагая одноименные разряды друг под другом, подвести под столбиком черту и напечатать сумму введенных чисел.

*Замечание*: наиболее часто применяемые типы вещественных чисел представлены короткими (4 байта) и длинными (8 байт) данными. Си использует для этой цели спецификаторы *float* и *double*.

Короткий вещественный формат по модулю обеспечивает представление чисел в диапазоне от 10-38 до 10+38 примерно с 7-8 значащими цифрами. Для 8-байтового формата диапазон существенно расширяется –от 10-308 до 10+308, а количество значащих цифр увеличивается до 15-16.

Для компактного вывода на каждое число вместе с его номером необходимо отвести столько позиций, чтобы длина строки была кратна ширине колонки. В противном случае часть последнего числа будет автоматически перенесена в следующую строку.

1. **Использование функций.** Известно, что любое четное число *N (N > 0)* может быть представлено в виде суммы двух простых чисел *(N = N1 + N2)*.Написать программу, которая отыскивает все такие разложения (о не единственности решения свидетельствует простейший пример: 4 = 1 + 3 = 2 + 2).

Алгоритм работы программы: она перебирает все возможные слагаемые, первое из которых не превосходит *0.5\*N*, и если оба они оказываются простыми, выводит полученный результат. Анализ слагаемых выполняется с помощью функции *prime*, которая возвращает значение «1», если число простое, «0», если составное.

int prime(long N)

{

long j;

if(N < 4) return 1;

if(N%2 == 0) return 0;

for(j=3; j\*j <= N; j+=2)

if(N%j==0) return 0;

return 1;

}

*Обратить внимание, где будет располагаться функция.*

*Осуществить проверку введённого значения*

1. **Преобразование типов, условные операторы.** «Вечный календарь». Даны натуральные числа d, m, g, которые обозначают число, месяц и год. Определить день недели, на который попадает введенная дата.

*Замечание*. Дата должна находиться в диапазоне от 1582 до 4902 гг. Номер дня недели (воскресенье - 0, понедельник - 1, ..., суббота - 6) равен остатку от деления на 7 значения выражения

[ 2.6 \* m - 0.2] + d + y + [y / 4] + [x / 4] - 2 \* x.

Если полученное значение будет отрицательным, для получения истинного значения необходимо прибавить 7.

d - номер дня в месяце (1, 2, ...);

m - номер месяца (корректировка введённых данных), нумерация начинается с марта (март - 1, апрель - 2,..., декабрь - 10, январь и февраль считаются месяцами с номерами 11 и 12 предыдущего года;

y - две младшие цифры года;

x - две старшие цифры года;

[a] - целая часть а.

Осуществить контроль ввода дат. Использовать операторы целочисленного деления и остатка от деления. При выводе наименования дня недели использовать оператор *switch*.

1. **Вложенные циклы.** «Отгадай число». Генерируется случайное число в диапазоне от 0 до 99. Задано количество попыток. Если пользователь угадывает число, программа выводит сообщение "правильно"; в случае несовпадения выдается подсказка (больше/ меньше введённое число по сравнению со сгенерированным числом). В случае совпадения выдаётся сообщении о количестве попыток. При несовпадении печатается сгенерированное число. По окончании игры выдаётся запрос: «Продолжить игру Y/N?». Первый цикл – «начать новую игру», выход из второго цикла – количество попыток исчерпано или число угадано.
2. **Обработка текстовой информации.** В программе вводится текст (признак конца вводав - CTRL/Z - в программе EOF). Считаем, что слова в тексте могут быть разделены символами пробел, табуляция, клавишей enter (количество пробельных символов произвольно). Все остальные знаки препинания (если они написаны слитно со словом) считать, что они относятся к слову (по желанию можно учитывать различные знаки препинания для выделения слов)

Написать программу:

* печатающую количество слов в тексте и гистограмму длин вводимых слов. Изобразить гистограмму в виде горизонтальных полос, напечатанных знаками "\*" (можно другими).

*Пример входных данных:*

*Язык Си был разработан сотрудником Bell Laboratories*

*(отделение известной телефонной компании AT&T)*

*Деннисом Ритчи*

В тексте 14 слов.

\*\*\*\*

\*\*

\*\*\*

\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*\*

\*\*\*\*\*

* печатающую количество предложений и количество слов в предложении (конец предложения - ". ? !"). Изобразить диаграмму в виде вертикальных полос, напечатанных знаками "\*" (можно другими).

*Пример.*

Пусть в тексте 4 предложения. В 1-м предложении ⎯ 4 слова, во 2-м ⎯ 3, в 3-м ⎯ 6, в 4-м ⎯ 5.

**\***

**\* \***

**\* \* \***

**\* \* \* \***

**\* \* \* \***

**\* \* \* \***

1. **Битовые операции.** Осуществить циклический сдвиг влево или вправо на заданное число позиций. Пользователь вводит число (целое или символ в зависимости от того, что сдвигают: integer или char), а также число позиций сдвига. В результате выполнения программы должно быть выведено на экран:

А) исходное число в десятичном и двоичном виде

Б) число, полученное в результате сдвига, в двоичном и десятичном формате.

Пример. Пусть исходное число 117.

Двоичное представление: 0111 0101.

1110101 = 117

1101011 = 107

1010111 = 87

0101111 = 47

1011110 = 94

Сдвиг вправо на 3 позиции: 1010 1011 = 17410.

Сдвиг влево на 4 позиции: 1011 1010 = 8710.

1. **Решить задачу о ранце, используя битовые операции.** Задаётся число предметов, их вес и полезность, а также грузоподъёмность ранца. Определить, какие предметы надо уложить в ранец, чтобы их суммарная стоимость была максимальной.

*Замечание.* Задача переборного типа. Перебор осуществляется по всевозможным представлениям двоичного числа. Количество комбинаций 2n, где n – количество предметов. Поиск комбинации, на которой суммарный вес предметов <= объёма ранца и суммарная стоимость предметов в ранце максимальная.

1. Написать программу, которая определяет, обладает ли введённая с клавиатуры целочисленная квадратная матрица размером n×n (n – нечётное число) симметрией:

* относительно горизонтали
* относительно вертикали
* относительно центра.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| \* | \* | \* | \* | \* | \* | \* |
| \* | \* | \* | \* | \* | \* | \* |
| \* | \* | \* | \* | \* | \* | \* |
|  |  |  |  |  |  |  |
| \* | \* | \* | \* | \* | \* | \* |
| \* | \* | \* | \* | \* | \* | \* |
| \* | \* | \* | \* | \* | \* | \* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| \* | \* | \* |  | \* | \* | \* |
| \* | \* | \* |  | \* | \* | \* |
| \* | \* | \* |  | \* | \* | \* |
| \* | \* | \* |  | \* | \* | \* |
| \* | \* | \* |  | \* | \* | \* |
| \* | \* | \* |  | \* | \* | \* |
| \* | \* | \* |  | \* | \* | \* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| \* | \* |  |  |  |  | \* |
| \* | \* |  |  |  |  | \* |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| \* |  |  |  |  | \* | \* |
| \* |  |  |  |  | \* | \* |

1. **Задача Джонсона. Обработки n деталей на 2 станках.**

Исходные данные:

n – количество деталей

T = ||tij|| - матрица времён обработки i –ой детали на j – м станке

Требуемый результат

1. Определить порядок обработки деталей, при котором суммарное время обработки будет минимальным.
2. Построить график Ганта – графическое изображение последовательности выполнения работ.

Алгоритм решения задачи (точный алгоритм):

* Ищем минимальный элемент в матрице (если таких несколько, то выбираем 1-ый).
* Если минимум в 1-ом столбце, то соответствующая деталь запускается в обработку первой (от уже имеющейся в последовательности обработки – от начала), если минимум во 2-м столбце, то соответствующая деталь запускается в обработку последней (от уже имеющейся в последовательности обработки – от конца).
* Эту деталь из рассмотрения исключаем.
* Алгоритм продолжается до тех пор, пока все детали не будут рассмотрены.

Пример: n = 6.

Матрица времён обработки:

|  |  |
| --- | --- |
| 7 | 1 |
| 3 | 5 |
| 1 | 2 |
| 6 | 4 |
| 3 | 3 |
| 3 | 2 |

Решение

|  |  |  |  |
| --- | --- | --- | --- |
| 1 деталь | 7 | 1 | Удалён на 1 шаге (конец) |
| 2 деталь | 3 | 5 | Удалён на 4 шаге (начало) |
| 3 деталь | 1 | 2 | Удалён на 2 шаге (начало) |
| 4 деталь | 6 | 4 | Удалён на 6 шаге (начало) |
| 5 деталь | 3 | 3 | Удалён на 5 шаге (начало) |
| 6 деталь | 3 | 2 | Удалён на 3 шаге (конец) |

Р = (3, 2, 5, 4, 6, 1)

График Ганта

1 станок: 32225554444446661111111

2 станок: 33 22222555 444466 1

Время обработки = 24

1. **Модификация задачи Джонсона.** 
   * Информация вводится из файла: в 1-ой строке – количество станков, в последующих строках – времена обработки соответственно на 1-м т 2- м станках (количество строк зависит от того, как осуществляется чтение из файла).
   * Предусмотреть динамическое выделение память для двумерного массива.
2. **Программа сортировки строк текста в алфавитном порядке.** Количество строк не известно. Определяется в программе по мере ввода. Признак конца ввода – пустая строка. Для доступа к отдельным строкам используется массив указателей. Каждый элемент массива указателей указывает на начало (на 1-ый символ) соответствующей строки. Строки при сортировке не меняются местами, а меняются значения в массиве указателей.

Необходимо реализовать 2 функции:

1. sort (любой метод сортировки) вызывается из функции main;
2. strcmp\_my (собственная функция реализует сравнение строк) вызывается из функции sort.

В функции main вводятся строки, и динамически выделяется память для каждой строки.

Предусмотреть проверку, выделена память или нет.

Результат выполнения программы:

* Печать исходного текста
* Печать отсортированного текста

1. **Рекурсивные функции.** Составить программу сортировки массива целых чисел в порядке возрастания. В программе должен быть реализован алгоритм сортировки слиянием, рекурсивную функцию которого надо модифицировать таким образом, чтобы для последовательностей длиной меньше 5 выполнялась сортировка вставками. Размер массива вводится в программе. Заполнение массива – случайными числами**.**

*Сортировка слиянием:* список разделяется на равные или практически равные части, каждая из которых сортируется отдельно. После чего уже упорядоченные части сливаются воедино:

1. массив рекурсивно разбивается пополам, и каждая из половин делиться до тех пор, пока размер очередного подмассива не станет равным единице;
2. выполняется операция алгоритма, называемая слиянием. Два единичных массива сливаются в общий результирующий массив, при этом из каждого выбирается меньший элемент (сортировка по возрастанию) и записывается в свободную левую ячейку результирующего массива.
3. Из двух результирующих массивов собирается третий общий отсортированный массив, и так далее. В случае если один из массивов закончиться, элементы другого дописываются в собираемый массив. В конце операции слияния, элементы перезаписываются из результирующего массива в исходный.

*Сортировка вставками:*

1. сортируемый массив можно разделить на две части — отсортированная часть и неотсортированная. В начале сортировки первый элемент массива считается отсортированным, все остальные — не отсортированные.
2. Начиная со второго элемента массива и заканчивая последним, алгоритм вставляет неотсортированный элемент массива в нужную позицию в отсортированной части массива. Таким образом, за один шаг сортировки отсортированная часть массива увеличивается на один элемент, а неотсортированная часть массива уменьшается на один элемент.
3. **Определяемые пользователями типы данных**. Используя структуры, создать простейшую базу данных, состоящую из одного файла. Структуру придумать самостоятельно.

В программе предусмотреть следующие функции:

* Создание новой БД.
* Чтение из файла уже существующей БД.
* Сохранение в файле.
* Ввод новых записей
* Поиск по ключу.
* Модификация.
* Просмотр.
* Удаление записи.

1. **Структуры данных – очередь.** Структура имеет поле fam, являющееся указателем на строку с фамилией и поле next, которое содержит указатель на следующий элемент. У последнего элемента этот указатель имеет значение NULL. По этому признаку определяется последний элемент очереди.

* Программа должна состоять из 3 файлов:
* заголовочный файл, содержащий объявление структуры и прототипы функций;

#define QUE struct queue

QUE { char \*fio;

QUE \*next;

};

extern void insert(QUE \*\* , char \*); //вставка в конец очереди

extern char \*take(QUE \*\*); / извлечение 1-го элемента из начала очереди;

* файл, содержащий описание функции работы с очередью (постановка в очередь нового элемента, извлечение 1-го элемента из начала очереди);
* файл, содержащий функцию main.

При добавлении нового элемента динамически выделяем память. При извлечении – освобождаем. В функции main создаётся очередь, а затем делится на 2 очереди: элементы с нечётными номерами остаются в 1-ой очереди, а с чётными переходят во 2-ую с сохранением порядка следования.

1. **Структуры данных – бинарное дерево.** Программа подсчета частоты встречаемых слов в тексте, читаемом из файла.

При моделировании деревьев их узлы представляют в виде структуры:

* поле для хранения информации
* поле с указателем на левое поддерево
* поле с указателем на правое поддерево

#define TREE struct tree

TREE { char \* wrd;

int count;

TREE \*left;

TREE \*right;

};

Для очередного слова динамически выделять память.