Для работы с символами используется тип char. Например, запомнить в переменной букву A можно так:

$$\text{char ch} =' A';$$

т.е. значение переменной ch делаем равной коду буквы A. Коды символов содержатся в специальных таблицах (ASCII), их значения обычно не важны. Распечатка значения с в формате *%c*: printf("%c",ch) (или putchar(ch)); ведет к выводу на экран буквы A. Коды имеют не только печатные символы, но и пробел, переход на новую строку ('\n'), табуляция ('\t') и некоторые другие управляющие символы. Функции для обработки символов содержатся в библиотеке <ctype.h>. Из полезных:

int isalpha(int); возвращает не нуль, если аргумент – буква английского алфавита, и 0 – иначе.

int isdigit(int); возвращает не нуль, если аргумент – десятичная цифра, и 0 – иначе,

int isprint(int); возвращает не нуль, если аргумент – печатаемый символ (в том числе пробел), и 0 иначе, управляющие символы печатными не являются;

int isspace(int); возвращает не нуль, если аргумент – пробельный символ (пробел, табуляция, '\t', возврат каретки, горизонтальной табуляции), и 0 иначе,

int islower(int); возвращает не нуль, если аргумент – символ нижнего регистра (прописная буква английского алфавита),

int isupper(int); возвращает не нуль, если аргумент – символ верхнего регистра (заглавная буква английского алфавита).

Замечание.

Коды десятичных цифр идут подряд в порядке возрастания, то же верно для кодов букв нижнего регистра, то же – для кодов букв верхнего регистра.

int putc(int ch, FILE *stream); записывает символ, содержащийся в младшем байте ch, в поток вывода, на который указывает stream. В случае успеха putc() возвращает записанный символ; в случае ошибки – EOF.

int putchar(int ch); выводит символ ch в stdout (на экран).

int getc(FILE *stream); возвращает следующий за текущей позицией символ во входном потоке stream и дает приращение указателю положения в файле.

При достижении конца файла getc() возвращает EOF.

Строка в Си – массив типа char * (т.е. последовательность символов), в котором есть символ с нулевым кодом ('\0' или просто 0). Все символы, которые стоят после нулевого байта, игнорируются. Константная строка заключается в двойные кавычки.

Для работы со строками есть библиотека <string.h> (нужно знать основные функции этой библиотеки). Функции работают со строками, ошибки сегментации не отслеживаются. Из основных:

size_t strlen(char *s); возвращает количество символов в строке s (конец строки, т.е. нулевой байт, в подсчете не участвует)

char *strcat(char *s1, const char *s2);

(контанкенация, или склейка): присоединяет s2 к s1, возвращает s1. Не проверяется корректность действия, т.е. есть ли в массиве s1 достаточное количество свободных ячеек для того, чтобы записать все символы s2. В ту ячейку, где находится конец строки 1, записывается начальный символ s2, в следующую ячейку – символ s2[1] и т.д., до конца строки s2 включительно. Если какие-то ячейки s1 и s2 являются общими (например, strcat(s,s+1)), никакой результат нельзя гарантировать (действие не определено, но программа скомпилируется).

char *strncat(char *s1, const char *s2, size_t n) подклеивает n начальных символов строки s2 к строке s1.

char *strcpy(char *s1, const char *s2); копирует строку s2 в строку s1, включая '\0', возвращает s1, работает с такими же ограничениями, как и strcat.

int strcmp(const char *s1, const char *s2); сравнивает s1 и s2 (лексикографически), возвращает значение 0, если строки одинаковы.

int strncmp(const char *s1,const char *s2, size_t n); сравнивает n байт в строках s1 и s2 (лексикографически), возвращает значение 0, если строки одинаковы.

char *strchr (const char *str, int ch); поиск символа ch в строке str. Возвращаемое значение: указатель на первое вхождение ch, если он найден в строке str, иначе NULL.

char *strstr(const char *strB, const char *strA); поиск строки strA в строке strB. Возвращаемое значение: NULL – если строка strA не входит в строку strB, иначе указатель на первое вхождение строки strA в строку strB (при сравнении не учитывается нулевой байт в строке strA, а только длина этой строки)

Для строк можно использовать функции для работы с произвольными областями оперативной памяти (в которых нет использования нулевого байта):

void *memcpy(void *dst, const void *src, size_t n); копирует из источника (src) в приемник (dst) n байтов. Такие же ограничения, как и для strcpy).

void *memset (void *dst, int ch, size_t n); n байт области памяти, на которую указывает dst, заполняются кодом символа, хранящимся в переменной ch. Так, char s[256]; memset(s,0,256); всю выделенную под массив s память заполняет символом конца строки. Для корректной работы с перекрывающимися областями памяти вместо memcpy используется функция

void *memmove(void *dst, const void *src, size_t n);

Результат работы такого кода:

char str[]="12345"; memcpy(str,str+1,strlen(str)); puts(str);

не определен (зависит от реализации), а такого:

char str[]="12345"; memmove(str,str+1,strlen(str)); puts(str);
вывод строки
2345
Более того, код

char str[]="12345"; memmove(str+1,str,strlen(str+1)); puts(str);

работает корректно, выводя строку
11234,
а код

char str[]="12345"; memcpy(str+1,str,strlen(str+1)); puts(str);

может зациклиться.

Ввод и вывод строк.

Для вывода строки в файл, помимо fprintf(FILE *,"%s", char *) можно использовать

fputs(const char *, FILE *in); Вывод на экран: puts(const char *) (после вывода на экран делает принудительный перевод на новую строку) или fputs(const char *, stdout) (без этого побочного эффекта).

Для ввода строки из файла используют функцию

char *fgets(char *str, int num, FILE *stream);

fgets считывает до num-1 символов из файла stream и помещает их в str. Символы считываются до тех пор, пока не встретится символ '\n', EOF или до достижения указанного предела (num-1). Далее в str после последнего считанного символа помещается нулевой конца строки.

В случае удачи fgets() возвращает str, при неудаче (количество прочитанных символов равно 0) возвращается NULL.

fgets считывает все символы, включая пробельные, и все их помещает в str. Для считывания строки с клавиатуры можно использовать fgets(str, num, stdin). В отличие от scanf("%s",str) она не будет игнорировать пробелы, ввод заканчивается нажатием клавиши Enter.

Слово – это максимальный по включению участок строки, не содержащий разделителей. Разделители – это некоторый набор символов, его помещают в строку разделителей. этой строки будет зависеть, на какие слова разбивается строка.

Слова из строки str можно выделять, просматривая str и проверяя, является ли текущий символ разделителей или нет (для этого можно использовать strchr(sep, *str), где sep – строка разделителей.

Для выделения слова также есть функция

char *strtok(char *str, const char *sep);, где str – разбиваемая строка, sep – строка разделителей.

Возвращаемое значение: NULL – если в строке str невозможно выделить слово, иначе – указатель на первый символ выделенного в строке слова. Если слово выделено, после него в str ставится конец строки. Для выделения следующего слова strtok вызывают с аргументами NULL, sep: strtok(NULL,sep);

Если набор разделителей – пробельные символы (пробел, табуляция, переход на новую стоку), для разбиения строки на слова можно использовать

int sscanf(char *buf, const char *format, arg-list) ;

Эта функция работает аналогично fscanf, но считывает информацию не из файла, а из строки. Отличие – в том, что при успешном считывании строка не будет сдвигаться (в отличии от файла, где указатель после считывании очередного числа, символа или строки передвигается на положение после прочитанного, что позволяет легко использовать fscanf в цикле для считывания следующей порции информации). Для считывании всех слов из строки нужно использовать формат $\%s\%n$:

sscanf(str,"$\%s\%n$",word,&k), где k – переменная целого типа, str - строка, из которой надо выделить слова, word – строка-приемник слова. Если возвращаемое значение равно 1 (слово найдено), то в word запишется найденное слово (с нулевым байтом), а в k – количество символов, просмотренных в строке str для выделения word (в k учитываются и пробельные символы). Все, что остается сделать для считывания следующего слова – увеличить str на k:

```
char s[]=" rtret rerer ttttt",*str=s,word[256]; int k;
for(;sscanf(str,"%s%n",word,&k)==1; str+=k)
puts(word);
```