

```

int codificare(int n)
{
    union { char ch[2];
           int i;
           } uniune;

    uniune.i = n;

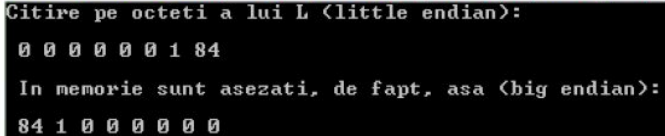
    uniune.ch[0] = uniune.ch[0]^uniune.ch[1];
    uniune.ch[1] = uniune.ch[0]^uniune.ch[1];
    uniune.ch[0] = uniune.ch[0]^uniune.ch[1];

    return uniune.i;
}

printf("Citire pe octeti a lui L (little endian):\n\n");
printf(" %d %d %d %d %d %d %d %d\n", *(ch+7), *(ch+6), *(ch+5), *(ch+4), *(ch+3), *(ch+2), *(ch+1), *ch);

printf("\n In memorie sunt asezati, de fapt, asa (big endian):\n\n");
printf(" %d %d %d %d %d %d %d %d\n", *ch, *(ch+1), *(ch+2), *(ch+3), *(ch+4), *(ch+5), *(ch+6), *(ch+7));

```



```

Citire pe octeti a lui L <little endian>:
0 0 0 0 0 0 1 84

In memorie sunt asezati, de fapt, asa <big endian>:
84 1 0 0 0 0 0 0

```

char* fgets(char *sir, int m, FILE *f)

- ❑ citește maxim m-1 caractere sau până la ‘\n’ și pune șirul de caractere în sir (adaugă la sfârșit ‘\0’).
- ❑ returnează adresa șirului citit.
- ❑ dacă apare vreo eroare întoarce NULL.

int feof(FILE *f)

f = pointer la FILE corespunzătoare fișierului pe care îl prelucrez.

funcția feof returnează 0 dacă nu s-a ajuns la sfârșitul fișierului la ultima operație de citire sau o valoare nenulă dacă s-a ajuns la sfârșitul fișierului.

int fseek(FILE *f, int nr_octeti, int origine)

```
void qsort(void *adresa, int nr_elemente, int  
dimensiune_element, int (*cmp)(const void *, const void *)) ;
```

```
int (*oper[4])(int a, int b) = {add, sub, mul, div};
```

```
void * malloc( int dimensiune);  
void * calloc( int numar, int dimensiune);  
void * realloc( void *p, int dimensiune);  
void* memcpy(void *d, const void* s, int n);  
void* memmove(void *d, const void* s, int n);  
void* memset(void *d, char val, int n);  
void* memchr(const void *d, char c, int n);
```

compararea a două tablouri pe octeți – funcția **memcmp**

- ❑ antet: `int memcmp(const void *s1, const void *s2, int n);`
- ❑ compară primii **n** octeți corespondenți începând de la adresele **s1** și **s2**.
- ❑ returnează 0 dacă octeții sunt identici, ceva mai mic decât 0 dacă **s1 < s2**, ceva mai mic decât 0 dacă **s1 > s2**

```
int fread(void *tablou, int dim_element, int nr_elem, FILE *f
```

- ❑ citește cel mult **nr_elem** elemente de dimensiune **dim_element** din fisierul referit de **f** la adresa **tablou**.

```
int fwrite(void *tablou, int dim_element, int nr_elem, FILE *f)
```

- ❑ scrie în fisierul referit de **f** cel mult **nr_elem** elemente de dimensiune **dim_element** de la adresa **tablou**;

conversia de la șir la un număr poate fi făcută cu ajutorul funcției **sscanf** și descriptori de format potriviți
exemplu:

```
char *string="-45.8614";  
double numar;  
sscanf(string, "%lf", &numar);  
printf("%f", numar);
```

conversia de la un număr la șir poate fi făcută cu ajutorul funcției **sprintf** și descriptori de format potriviți
exemplu:

```
char string[12];  
int numar=897645671;  
sprintf(string, "%d", numar);  
printf("%s", string);
```

void free(void *p);

sunt în fișierul ctype.h

Prototip	Descriere
int isdigit(int c)	Returnează true dacă c este cifră și false altfel
int isalpha(int c)	Returnează true dacă c este literă și false altfel
int islower(int c)	Returnează true dacă c este literă mică și false altfel
int isupper(int c)	Returnează true dacă c este literă mare și false altfel
int tolower(int c)	Dacă c este literă mare, tolower returnează c ca și literă mică. Altfel, tolower returnează argumentul nemodificat
int toupper(int c)	Dacă c este literă mică, toupper returnează c ca și literă mare. Altfel, toupper returnează argumentul nemodificat
int isspace(int c)	Returnează true dacă c este un caracter <i>white-space</i> — <i>newline</i> ('\n'), <i>space</i> (' '), <i>form feed</i> ('\f'), <i>carriage return</i> ('\r'), <i>horizontal tab</i> ('\t'), <i>vertical tab</i> ('\v') — și false altfel