

Introduction to Programming

Important Dates

Endterm Test

- ▶ 2nd December 2022 (Friday)
- ▶ 2–4 pm, IK–201 OR 4–6 pm, IK–201

Retake Midterm Test

- ▶ 9th December 2022 (Friday)
- ▶ 2–4 pm, IK–201

Retake Endterm Test

- ▶ 9th December 2022 (Friday)
 - ▶ 4–6 pm, IK–201
- 

QSORT function – stdlib.h

```
void qsort (void* a, size_t n, size_t size,  
            int (*compar)(const void*,const void*));
```

- ▶ Sorts the **n** elements of the array pointed to by **a**, each element **size** bytes long, using the **compar** function to determine the order.
- ▶ The sorting algorithm used by this function compares pairs of elements by calling the specified **compar** function with pointers to them as argument.
- ▶ The function does not return any value, but modifies the content of the array pointed to by **a** reordering its elements as defined by **compar**.
- ▶ The order of equivalent elements is undefined.

QSORT function – stdlib.h

```
void qsort (void *a, size_a n, size_a size),  
int(*compar)(const void* a1,const void* a2) );
```

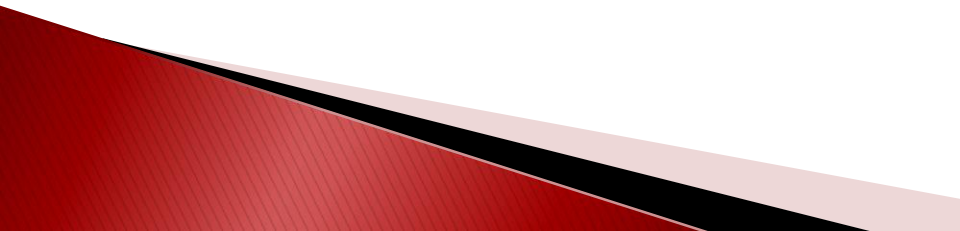
- **a**: Pointer to the first object of the array to be sorted, converted to a void*.
- **n**: Number of elements in the array pointed to by **a**.
- **size**: Size in bytes of each element in the array.
- **compar**: Pointer to a function that compares two elements.
- This function is called repeatedly by qsort to compare two elements.

Compar function

- ▶ Pointer to a function that compares two elements.
- ▶ This function is called repeatedly by qsort to compare two elements.
- ▶ **int compar(const void* a1, const void* a2);**
- ▶ Taking two pointers as arguments (both converted to const void*).
- ▶ The function defines the order of the elements by returning.
- ▶ Return value meaning:
 - **<0** The element pointed to by a1 goes before the element pointed to by a2.
 - **0** The element pointed to by a1 is equivalent to the element pointed to by a2.
 - **>0** The element pointed to by a1 goes after the element pointed to by a2.

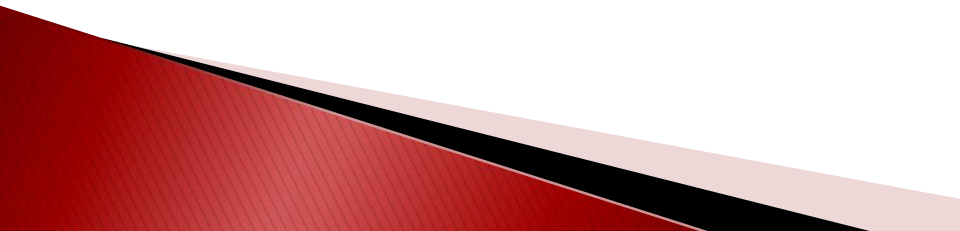
Compar function

```
int compar(const void *a1, const void *a2)
{
    int *a = (int *) a1;
    int *b = (int *) a2;
    if (*a < *b)
        return -1; //decreasing order 1
    if (*a == *b)
        return 0;
    if (*a > *b)
        return 1; //decreasing order -1
}
```



Compar function

```
int compar(const void *a1, const void *a2)
{
    int a = *(int*) a1;
    int b = *(int*) a2;
    if (a < b)
        return -1;
    if (a == b)
        return 0;
    if (a > b)
        return 1;
}
```



Compar function

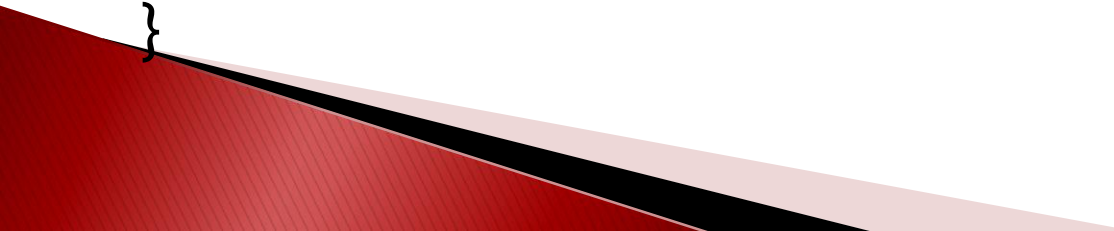
```
int compar(const void *a1, const void *a2)
{
    return ( *(int*) a1 - *(int*) a2);
}
```


QSORT

```
int main()
{
    srand(time(NULL));
    int a[50], i;
    for (i = 0; i < 50; i++)
        a[i] = rand()%51 + 20;
    print_out(a, 50);

    qsort(a, 50, sizeof(int), compar);

    print_out(a, 50);
    return 0;
}
```



Exercise

- ▶ Sort the following array, which contains real numbers, using the **QSORT** function!
- ▶ `double a[7] = { 863, -37, 54, 9520, -3.14, 25, 9999.99 };`

Solution

```
void print_out(double a[], int n)
```

```
{  int i;
    for (i = 0; i < n; i++)
        printf("%.2lf ", a[i]);
    printf("\n");
}
```

```
int compar(const void *a1, const void *a2)
```

```
{
    double a = *(double *) a1;
    double b = *(double *) a2;
    if (a < b)
        return -1;
    if (a == b)
        return 0;
    if (a > b)
        return 1;
}
```

```
int main()
```

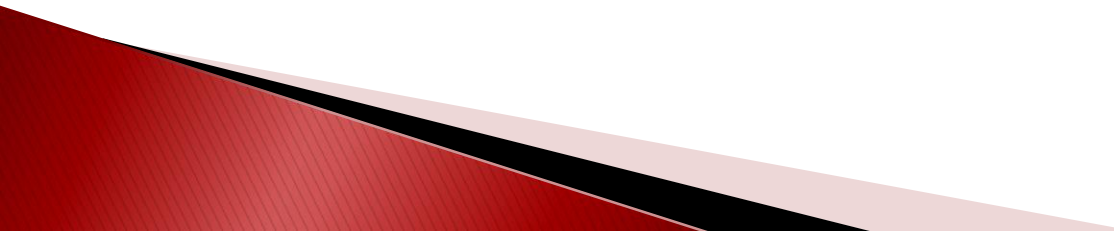
```
{
    double a[7] = { 863, -37, 54, 9520, -3.14,
                    25, 9999.99 };
    printf("Before sorting:\n");
    print_out(a, 7);
```

```
    qsort(a, 7, sizeof(double), compar);
```

```
    printf("After sorting:\n");
    print_out(a, 7);
```

```
    return 0;
}
```

Strings

- ▶ strings are a series of characters
 - ▶ can be stored in character arrays
 - ▶ we use statically and dynamically allocated character arrays
 - ▶ every string is closed by the **EOS** (End Of String) or null character `'\0'`
- 

Declaration

Strings have two types "pointer to char" (char *) or array of char.

- ▶ **char s1[256];**
 - in s1 up to 255 long strings can be stored, the value of the elements is undefined.
- ▶ **char s2[10] = {'a', 'p', 'p', 'l', 'e', '\0'};**
 - s2 stores the string "apple" . The value of the remaining elements is undefined.
- ▶ **char s3[10] = "apple";**
 - Equivalent of the previous one but is more compact and readable. Not the array but its elements are given value.
- ▶ **char s4[10] = {'a', 'p', 'p', 'l', 'e'};**
 - Does not store a string, unless the implementation nullifies the remaining elements of the array.

Input strings

- ▶ **scanf("%s", s);**
 - !!! no **&** character before the variable name
 - we can input a string which does not contain space.
- ▶ **fgets(s, n, stdin);**
 - fgets reads at most the next $n-1$ characters into the array **s**, stopping if a newline is encountered; the newline is included in the array, which is terminated by `'\0'`.
 - fgets returns **s** or NULL if end of file or error occurs.
- ▶ **gets(s);**
 - input a string until ENTER.

Output strings

- ▶ **printf("%s", s);**
- ▶ **fputs(s, stdout)**
 - fputs writes the string s (which need not contain \n) on stream;
 - it returns nonnegative, or EOF for an error.
- ▶ **puts(s);**
 - puts writes the string s and a newline to stdout.
 - it returns EOF if an error occurs, non-negative otherwise.

Exercise

- ▶ Write a program which inputs a word up to 32 characters and prints its first character.

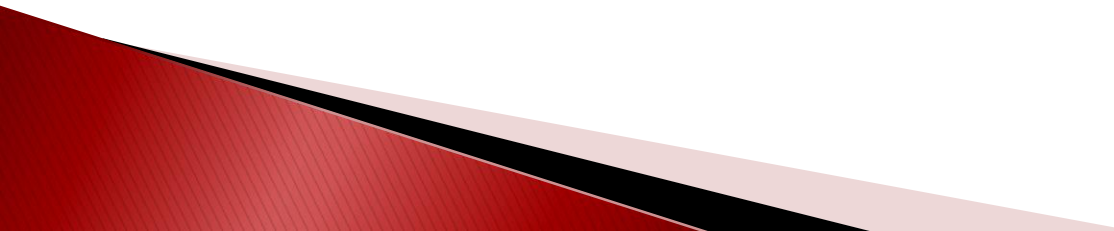
- ▶ **Example**

Input: apple

Output: a

Solution

```
char s[32];  
scanf("%s", s);  
printf("%c\n", s[0]);
```



Character Identification

<ctype.h>

isalnum(c) – Check if character is alphanumeric

isalpha(c) – Check if character is alphabetic

isblank(c) – Check if character is blank

iscntrl(c) – Check if character is a control character

isdigit(c) – Check if character is decimal digit

isgraph(c) – Check if character has graphical representation

islower(c) – Check if character is lowercase letter

isprint(c) – Check if character is printable

ispunct(c) – Check if character is a punctuation character

isspace(c) – Check if character is a white-space

isupper(c) – Check if character is uppercase letter

isxdigit(c) – Check if character is hexadecimal digit

tolower(c) – Convert uppercase letter to lowercase

toupper(c) – Convert lowercase letter to uppercase

Exercise

Write a program which inputs a word and transforms the letters:

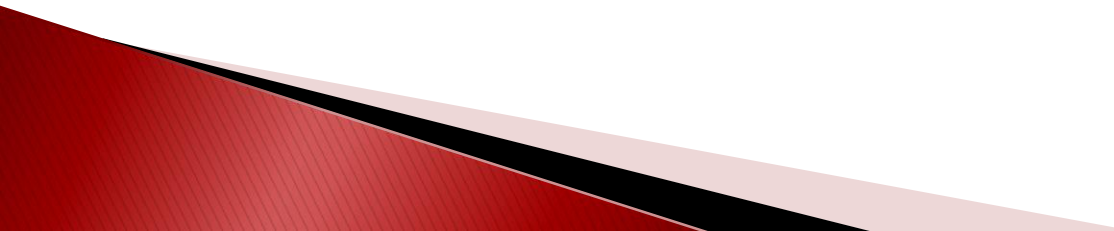
- a) lowercase letter → uppercase letter
- b) uppercase letter → lowercase letter

Example: ApplE – aPPLe

strlen(s) – length of the string
#include <string.h>

Solution

```
int i;  
char s[50];  
scanf("%s", s); //fgets(s,50,stdin);  
    for (i=0; i< strlen(s); i++)  
    {  
        if (isupper(s[i]))  
            s[i]=tolower(s[i]);  
        else  
            if (islower(s[i]))  
                s[i]=toupper(s[i]);  
    }  
printf("%s\n",s); //fputs(s,stdout);
```

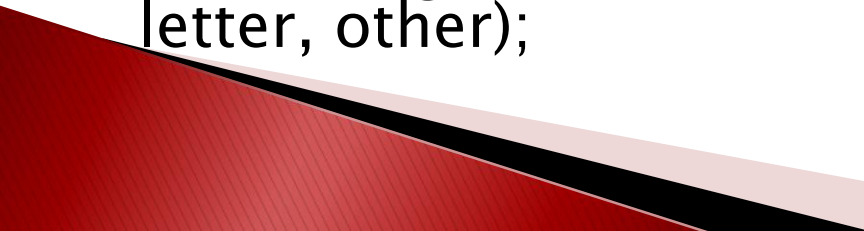


Exercise

Write a program which counts the letters, numbers and other characters which the input string contains.

Solution

```
int i, digit=0, letter=0, other=0;
char s[50];
fget(s,50,stdin);
    for (i=0; i< strlen(s)-1; i++)
    {   if (isdigit (s[i]))
            digit++;
        else
            if (isalpha(s[i]))
                letter++;
            else
                other++;
    }
printf("Digits=%d\nLetters=%d\nOthers=%d\n", digit,
letter, other);
```



String manipulation

<string.h>

strlen() This function returns a type **int** value, which gives the length or number of characters in a string, not including the **NULL** byte end marker.

Example:

- ▶ **int** len;
- ▶ **char** string[256]="apple";
- ▶ **len** = strlen(string); -> 5

strcat() This function "concatenates" two strings: that is, it joins them together into one string.

The effect of:

- ▶ **char** *new,*this, onto[255];
- ▶ **new** = strcat(onto,this);

String manipulation

<string.h>

strcpy() This function copies a string from one place to another. Use this function in preference to custom routines: it is set up to handle any peculiarities in the way data are stored.

Example:

- ▶ **char *to,*from;**
- ▶ **to = strcpy (to,from);**

Where **to** is a pointer to the place to which the string is to be copied and **from** is the place where the string is to be copied from.

String manipulation

<string.h>

- ▶ **strcmp()** This function compares two strings and returns a value which indicates how they compared.

Example:

- ▶ **int** value;
- ▶ **char** *s1,*s2;
- ▶ value = **strcmp**(s1,s2);

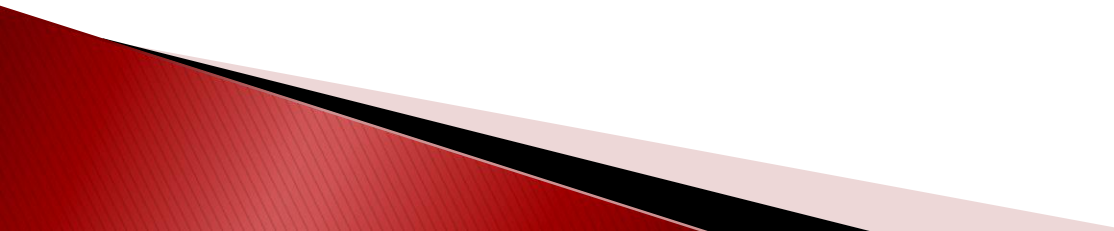
- ▶ The value returned is:
 - 0 if the two strings were identical.
 - > 0, if s1 > s2, alphabetically
 - < 0, if s1 < s2, alphabetically

Exercise

- ▶ Write a program which reverses the input string.
- ▶ **For example:**
 - cat → tac
 - people → elpoep
 - informatics → scitamrofni

Solution

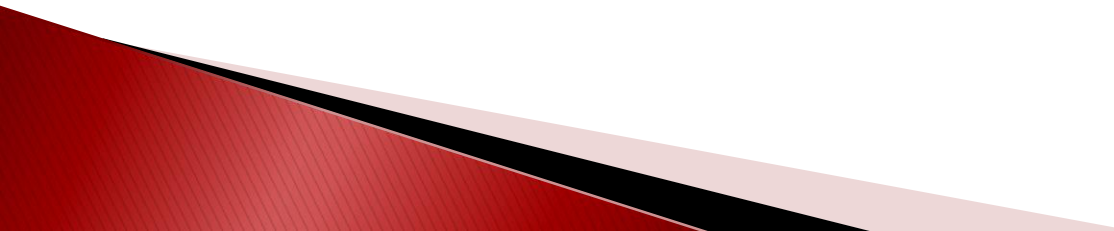
```
char s[20], c;  
int i, j;  
scanf("%s",s);  
for( i=0, j=strlen(s) - 1; i<j; i++, j--)  
{  
    c=s[i];  
    s[i]=s[j];  
    s[j]=c;  
}  
printf("%s",s);
```



Exercise

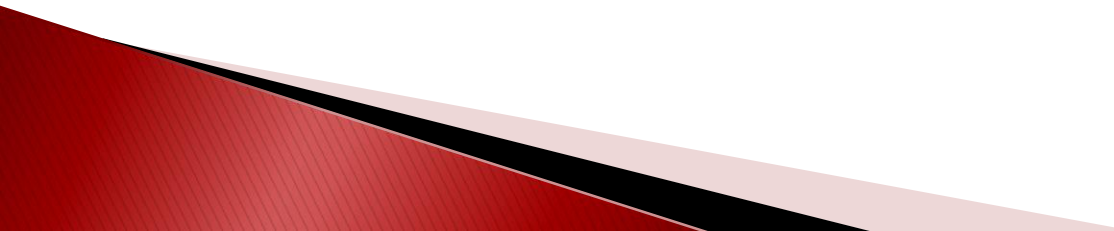
- ▶ Write a program which defines if the input string is a palindrome or not.

Example:

- eye, level, madam, radar
 - no lemon no melon
 - Cain a maniac.
 - A Toyota. Race fast, safe car. A Toyota.
- 


Solution – 1

```
char s[50];  
int i, len, c=0;  
scanf("%s",s);  
len=strlen(s);  
for(i=0;i<len;i++)  
    if (s[i]==s[len-i-1])  
        c++;  
if (c==len)  
    printf("Palindrom!\n");  
else  
    printf("Not Palindrom!\n");
```



Solution -2

```
char s[50], r[50],c;  
int i, j;  
printf("word=");  
scanf("%s",s);  
strcpy(r,s);  
for(i=0, j=strlen(r)-1; i<j; i++, j--)  
{  
    c=r[i];  
    r[i]=r[j];  
    r[j]=c;  
}  
if (strcmp(s,r)==0)  
    printf("Palindrom!\n");  
else  
    printf("Not Palindrom!\n");
```



Exercise – Qsort

Sort the following characters of the string.
Use the **qsort** function!!

▶ **char s[50]="It's a beautifull day!";**

Solution

```
int compar1(const void *a1, const void *a2)
{
    char a= *(char *) a1;
    char b = *(char *) a2;
    if (a < b)
        return -1;
    if (a == b)
        return 0;
    if (a > b)
        return 1;
}

int main(){
    char s[50]="It's a beautifull day!";
    qsort(s, strlen(s), sizeof(char), compar1);
    puts(s);
    return 0;
}
```


Solution

```
int compar2(const void *a1, const void *a2)
{
    return *(char *) a1 - *(char *) a2;
}
```

```
int compar3(const void *a1, const void *a2)
{
    return strcmp((char*)a1, (char*)a2);
}
```

