ARRAYS OF CHARACTERS (ASCIIZ arrays)

In the C programming language there is no separate data type for a text (string of characters). Instead, specialized arrays of the char type are used to represent the strings. The final element of such an array is a so-called zero-terminator, i.e. a control character **\0**. Thus, character arrays are often called **ASCIIZ strings**.

Example:

```
#include <stdio.h>
#include <string.h>
main()
{
  char      A[256];   /* max string lenght is 256 symbols */
  char      B[11];
  char      C[24];
  strcpy (A,"IBM PC Pentium");
  strcpy (B,"Windows 2000");
  strcpy (C,"");        /* cleaning a string */
  printf ("A= %s\n", A);
  printf ("B= %s\n", B);
  strcpy (C, B);
  printf ("C= %s\n",C);
}
```

Here character arrays (strings) are defined: A, B, C. The *strcpy* command allows initialization of theses arrays.

Being arrays, strings can be addressed by using pointers, for example:

```
#include <stdio.h>
#include <string.h>
main()
{
  char *message;
  message = " Hello Students";
  puts(message);
}
```

Here, **message* is a pointer to character, the assignment statement *message = "Hello Students"* assigns the address of the first byte of this character string to the *message* variable. Then, *puts* outputs all the characters (up to the zero character at the end of the string).

Main standard functions for working with strings:

strlen() – string length*;*
strcat() – combines two strings (concatenation);

strcmp() – compares string contents;
strcpy() – copies strings.

Example:

```
#include <string.h>
#include <stdio.h>
main()
{
  char k[60]="Mano batai buvo du";
  char l[20]="Vienas dingo - nerandu";
  printf("Eilutės ilgis= %d\n ",strlen(k));
  strcat(k,l);
  puts(k);
}
```

STRUCTURES (Records)

Unlike an array, a structure allows programmer to combine variables of different data types into a single new data type, i.e. **struct**:

```
#include <stdio.h>
#include <string.h>

typedef  struct student {
      char  namefname[40];
      int    year;
      int    group;
};
main()
{
struct student A, B;
strcpy(B. namefname,"J.Ivanauskas"); B.year = 2010;
B.group = 1;
printf("Name, family name = %s\n", B.namefname);
printf("Year = %d\n", B.year);
printf("Group = %d\n",B.group);
}
```

In C/C++, a structure may consist of other data structures. For example, an array *st_groups* defined below is made up of *student* structures (records):

```
#include <stdio.h>
#include <string.h>

typedef  struct student {
      char  namefname[40];
      int    year;
      int    group;
};
main()
{
  struct student st_groups[30];
  strcpy(st_group[1].namefname,"J.Ivanauskas");
  st_groups [1].year = 2000;
  st_groups [1].group = 1;
  strcpy(st_groups [2].namefname,"P.Petrauskas");
  st_groups[2].year = 2000;
  st_groups[2].group = 2;
  printf("Name, surname = %s\n", st_groups [1].vpavarde);
  printf("Year = %d\n", st_groups [1].year);
  printf("Grupė = %d\n", st_groups [1].group);
  printf("Name, surname = %s\n", st_groups [2].namefname);
  printf("Year = %d\n", st_groups [2].year);
  printf("Group = %d\n", st_groups[2].group);
}
```