Arrays of char

Operations. String processing functions.
Dynamic memory allocation.

array of char

Declarare:

```
char  <nume array> [<diapazon>];
```

Exemple

```
char x[100], y[100];
```

```
   char *ptr;
   int n;

   ...
// memory allocation
   ptr = (char*)calloc(n, sizeof(char));

// value allocation
   ptr = ”something”;
```

Exemple

bin_indexed_tree.c | string_examples_001.c

```c
#include <stdlib.h>
#include <stdio.h>

struct word {char *cuv;} prop[20];

int main()
{
    int n = 3, i;
    for (i=1; i<=n; i++)
    {
        prop[i].cuv = (char*)calloc(20, sizeof(char));
        scanf("%s", prop[i].cuv);
    }

    printf("\n Entered names are: \n");
    for (i = 1; i<=n; i++)
     printf("%s\t", prop[i].cuv);
    return 0;
}

```

Select E:\Work\UTM\Programarea_C

```
Valeriu
Isaak
Jack

 Entered names are:
Valeriu Isaak   Jack
--------------------------------
Process exited after 45.42 s
Press any key to continue .
```

Task 1

The program from previous slide ask for a memory block of 3 x 20 bytes. Rewrite the program to use a number of bytes to fit entered words EXACTELY. Do not use some standard functions from **string** or other libraries!

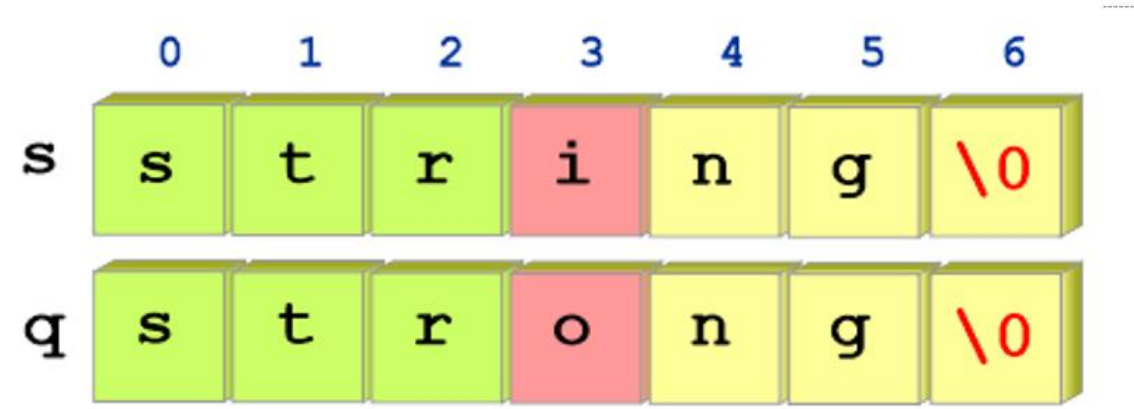
Some examples:

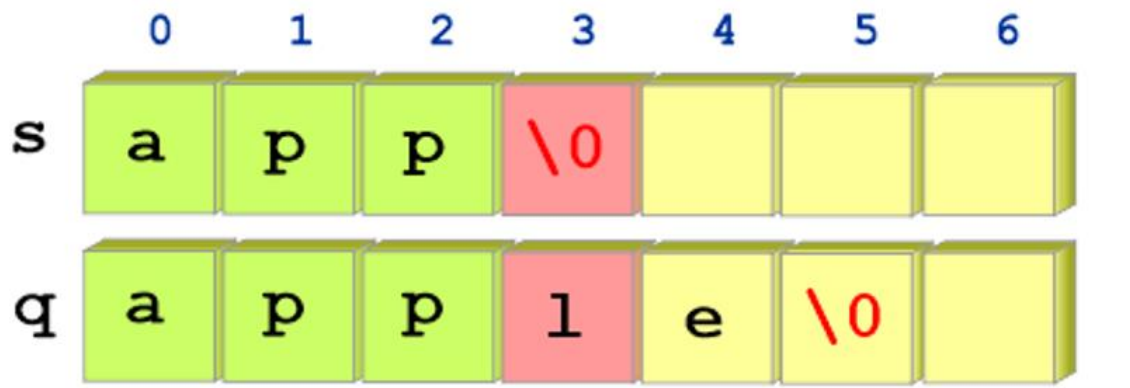
String length

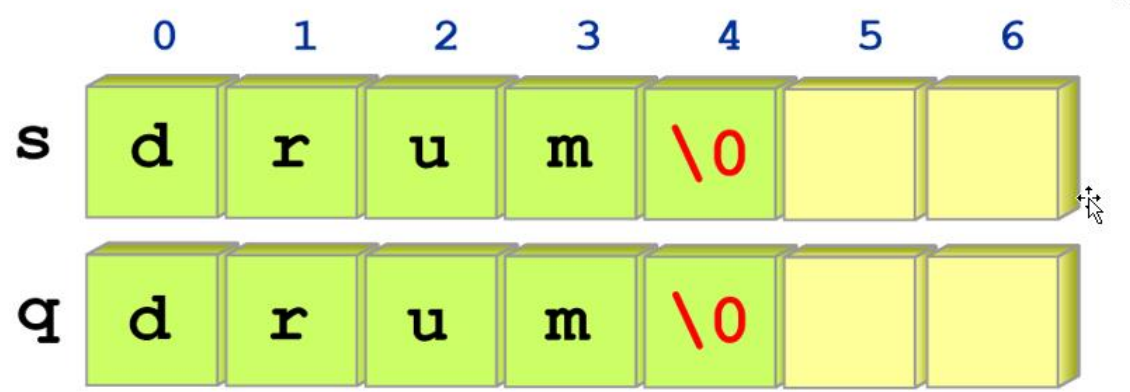
```
int lungimesir(char *a)
{
 int i = 0;
 while (a[i] != '\0' ) i++;
 return i;
}
```

Char to mark
END OF STRING

Strings comparation


0 1 2 3 4 5 6
s
s t r i n g \0
q
s t r o n g \0


0 1 2 3 4 5 6
s
a p p \0
q
a p p l e \0


0 1 2 3 4 5 6
s
d r u m \0
q
d r u m \0

Some examples:

Compare Strings

```
int compstrings (char *a, char *b)
{
 int i = 0;
 while (a[i]== b[i] && a[i]!='\0' && b[i]!='\0') i++;
 if (a[i] =='\0' && b[i] =='\0') return 0;
 if (a[i] =='\0' && b[i] !='\0') return -1;
 if (a[i] !='\0' && b[i] =='\0') return 1;
 if (a[i] !='\0' && b[i] !='\0')
     if (a[i] > b[i] ) return 1; else return -1;
}
```

More string operations

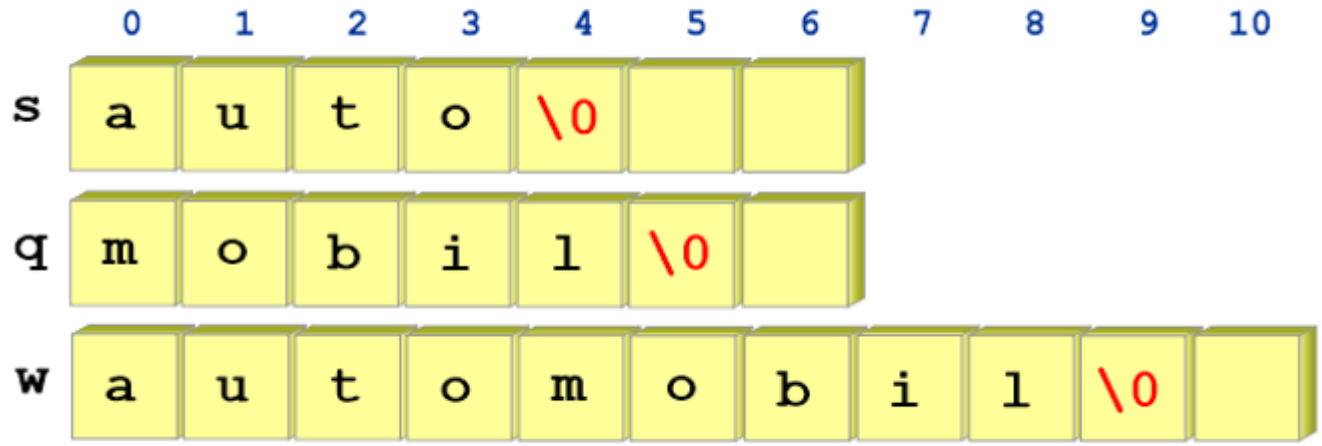
- Concatenation

Concatenation

string + string

=

stringstring!



To create the string w - the concatenation of the strings s and q :

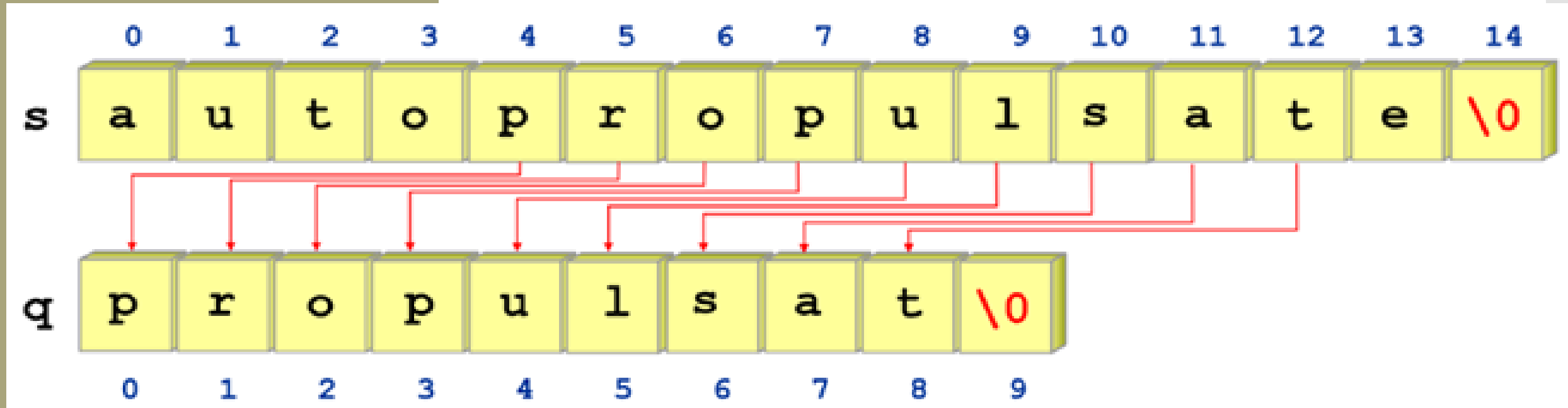
1. Define the string w with a number of elements, which is equal to or exceeds the sum of the lengths of the strings s and q.
2. Add consecutively in w, starting with the index 0, all the elements from s.
3. The value k of the index of the last added character is retained.
4. The characters of **q** are added consecutively in w, starting with w[k+1].
5. Fix the end of the string by adding the character '\ 0'.

Code

```
#include <iostream>
    using namespace std;

void mystrconcat(char *s, char *q, char *w)
{
    int i = 0; while (s[i] != '\0') { w[i] = s[i]; i++;}
    int j = 0; while (q[j] != '\0') { w[i] = q[j]; i++; j++;}
   // w[i] = '\0';
}

int main()
{
    char q[20] = {"Auto"};
    char s[20] = {"Basculanta"};
    char z[40];
    mystrconcat(q, s, z);
    cout << z << endl;
    mystrconcat(s, q, z);
    cout << z << endl;
    return 0;
}
```

Code v2

```
#include <iostream>
    using namespace std;
char *mystrconcat(char *s, char *q)
{
    int i = 0; while (s[i] != '\0') i++;
    int j = 0; while (q[j] != '\0')
        { s[i] = q[j]; i++; j++;}
    return s;
}
int main()
{
    char q[20] = {"Auto"};
    char s[20] = {"Basculanta"};
    cout << mystrconcat(q, s) << endl;
    return 0;
}
```



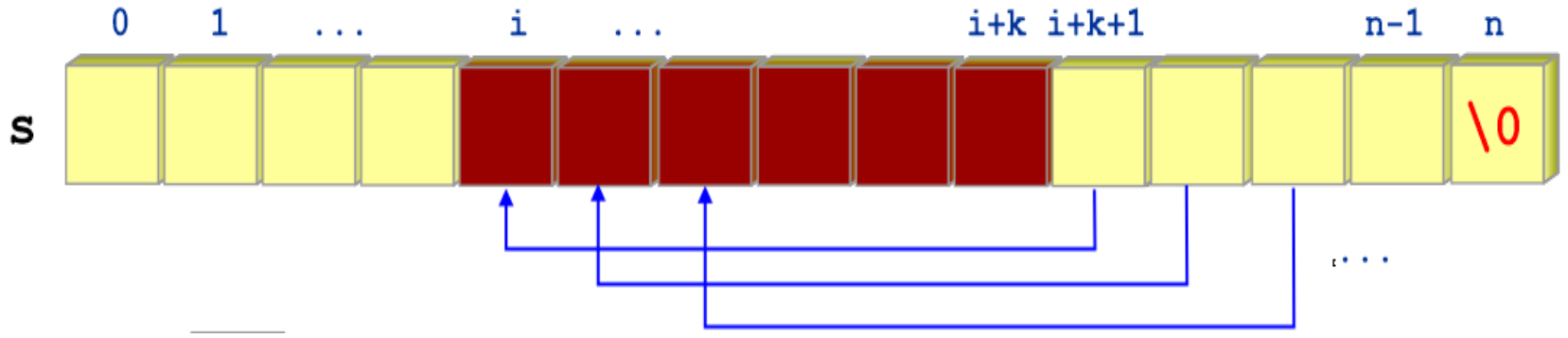
(Sub)String Copy

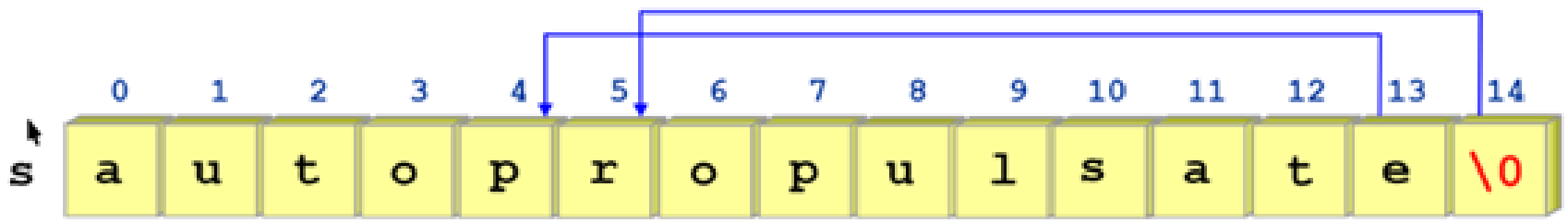
```
#include <iostream>
    using namespace std;

void mystrcpy(char *s, int i, int k, char *q)
{   int j;
    for ( j = i; j < i + k; j++)
        if (s[i] !='\0') q[j - i] = s[j];
        else break;
    q[j - i] = '\0';
}

int main()
{
    char s[40] = {"autopropulsata"};
    char z[40];
    mystrcpy(s, 4, 9, z);
    cout << z << endl;
    mystrcpy(s, 4, 15, z);
    cout << z << endl;
    return 0;
}
```

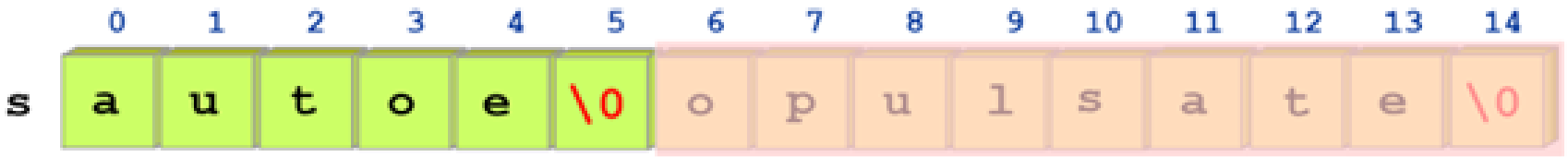
```
#include <iostream>
    using namespace std;

char *mystrcpy(char *s, int i, int k)
{   char *q;
    q = (char *)malloc((k-i+2)*sizeof(char));
    int j;
    for ( j = i; j < i + k; j++)
        if (s[i] !='\0') q[j - i] = s[j];
        else break;
    q[j - i] = '\0';
    return q;
}
int main()
{
    char s[40] = {"autopropulsata"};
    char *z;
    z = mystrcpy(s, 4, 5);
    cout << z << endl;
    cout << mystrcpy(s, 4, 15) << endl;
    return 0;
}
```

Substring Removal

The Scheme


0 1 ... i ... i+k i+k+1 n-1 n
s
\0


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
s
a u t o p r o p u l s a t e \0


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
s
a u t o e \0 o p u l s a t e \0

```
#include <iostream>
    using namespace std;

void mystrem(char *s, int i, int k)
{   int j;
    for ( j = 0; j < i; j++)
       if (s[j] =='\0') return; // s sfarseste inainte
    for ( j = i; j < i + k; j++) // de a lichida ceva
        if (s[j] =='\0')
        { s[i] = '\0'; return; } // s sfarseste
    int p = i;                   // fara concatenare
    while (s[j] != '\0')         // lipim fragmentul
        {                        // ramas la dreapta
            s[p] = s[j];
            p++; j++;
        }
    s[p] = '\0';
}
```

```
int main()
{
    char s[40] = {"autopropulsata"};
    char z[40] = {"autopropulsata"};
    char q[40] = {"paracetamol"};
    mystrem(s, 15, 3);
           // nimic de radiat
    cout << s << endl;
    mystrem(z, 6, 12);
           // radiere pana la sfarsit
    cout << z << endl;
    mystrem(q, 3, 6);
           // radiere pe interior
    cout << q;
    return 0;
}
```

Convert Integer to a string

The method

To convert a number into a string,

first separate the digits of the number. Start with the rightmost digit - the units digit:

```
while(n)
       {  k = n % 10;  n = n / 10;  }
```

Second: convert k to a char and add it to a string.

```
i = 0;
    while(n)
       {      k = n % 10; n = n / 10;
              s[i] = k +'0'; i++;
       }
```

Third: reverse the string:

```
for (int j = 0; j < i / 2; j++)
   {
      char q = s[j];  s[j] = s[i - 1 - j]; s[i - 1 - j] = q;
   }
```

```
#include <iostream>
    using namespace std;
long n;
char s[100];

void myitoa(long n, char *s)
{
    int i = 0, k;
    while(n)
              {  k = n % 10;  n = n / 10;   s[i] = k +'0';    i++;   }
    for (int j = 0; j < i / 2; j++)
              {  char q = s[j]; s[j] = s[i - 1 - j];  s[i - 1 - j] = q;  }
}
int main()
{
    scanf("%ld", &n);    myitoa(n,s);    cout << s;    return 0;
}
```

Convert a string to an integer

Pas 0. **i ← 0 , n ← 0**.

Pas 1. Dacă **s[i] = '\0'** trecem la Pas 4, altfel – la Pas 2.

Pas 2. // verificare a validității caracterului curent

Dacă **s[i]∈ {'0', ...,'9'}** trecem la Pas 3 altfel – STOP, **return -1**.

Pas 3. // transformarea caracterului în valoare numerică și "lipirea" de fragmentul deja existent

k ← s[i] - '0'. // Calculăm valoarea cifrei curente

n ← n x 10 + k // Cifrele din număr sunt deplasate cu o poziție spre stânga,

// k ocupă poziția unităților

i ← i+1, revenim la Pas 1

Pas 4. Returnăm **n**.

Algorithm

```
#include <iostream>
    using namespace std;
int n;

int myatoi(char *s)
{
    int i = 0, n = 0;
    while(s[i] != '\0')
    {
        if (s[i] < '0' || s[i] > '9') return -1;
        n = n * 10 + (s[i] - '0');  i++;
    }
    return n;
}
int main()
{   char s[10] = {"2300056"};      n = myatoi(s);    cout << n << endl;
    char q[10] = {"00056"};        n = myatoi(q);    cout << n << endl;
    char z[10] = {"1000#56"};      n = myatoi(z);    cout << n;
    return 0;
}
```

Reminder: how to read strings

Citire șir de caractere:

- **`scanf`**`() (fără spații!)`
- **`gets`**`()`

Thanks'!

Working with chars.