**strcpy**

Funkcja kopiuje łańcuch znaków.

char \*strcpy(char \*dest, const char \*src)

{char\* p=dest;while (\*dest++ = \*src++);

return p;}

**strcat**

Funkcja dołącza łańcuch.

char \*strcat(char \*dest, const char \*src)

{char\* p=dest;while (\*dest) dest++; /\* przejdź na koniec łańcucha docelowego \*/while (\*dest++ = \*src++); /\* dołącz łańcuch źródłowy \*/return p;}

Jest to równoważne: strcpy(dest[strlen(dest)+1], src);

**Strchr**

Funkcja zwraca wskaźnik do pierwszego wystąpienia znaku c, jeśli nie zostanie

on znaleziony wynikiem funkcji jest NULL.

char \*strchr(const char \*s, int c)

{char\* temp = (char\*)s;

while (\*temp && \*temp != c) temp++;

if (\*temp == '\0')return NULL;return temp;}

**strcmp**

Funkcja zwraca różnicę kodów ASCII pierwszego różniącego się znaku. Jeśli porównywane łańcuchy są identyczne zwraca 0.

int strcmp(const char \*s1, const char\*s2)

{while ((\*s1) && (\*s2) && (\*s1==\*s2)) s1++, s2++;

return (int)\*s1-(int)\*s2;}

Aby wszystko było w porządku należałoby zrzutować s1 i s2 na char\*, gdyż obie

zmienne są stałymi wskaźnikami. Nie chcę zaciemniać kodu zbędnymi zmiennymi,

poza tym kompilator może rzucić co najwyżej ostrzeżeniem.

**strcspn**

Funkcja zwraca indeks pierwszego znaku z s1 należącego do zbioru znaków set.

size\_t strcspn(const char \*s, const char \*set)

{size\_t pos = 0;char \*temp;

while (\*s) /\* przetwarzaj wszystkie znaki łańcucha \*/{temp = (char\*)set;

while (\*temp) /\* porównaj każdy znak ze zbioru z bieżącym znakiem \*/

if (\*s == \*temp) return pos; /\* zwróć indeks bieżącego znaku gdy należy on

do zbioru \*/else temp++;s++;pos++;}

return pos;}

**strlen**

Funkcja zwraca długość łańcucha, nie uwzględnia kończącego zera.

size\_t strlen(const char\* string)

{size\_t len = 0;

while (\*string++) len++;return len;}

**strncat**

Funkcja dołącza maksymalnie n znaków.

char\* strncat(char \*s1, const char \*s2, size\_t n)

{size\_t num = 0;char\* result = s1;while (\*s1) s1++;

while ((\*s1++ = \*s2++) && (++num < n));

\*s1 = 0;return result;}

**strncmp**

Funkcja porównuje maksymalnie n znaków.

int strncmp(const char \*s1, const char \*s2, size\_t n)

{size\_t num = 0;

while ((\*s1) && (\*s2) && (\*s1==\*s2) && (++num < n)) s1++, s2++;return (int)\*s1-(int)\*s2;}

**strncpy**

Funkcja kopiuje maksymalnie n znaków.

>char\* strncpy(char \*s1, const char \*s2, size\_t n)

{char\* result = s1;size\_t num = 0;

while ((\*s1++ = \*s2++) && (++num < n));

\*s1 = 0; /\* dopisz kończące zero - w standardzie tego nie ma! \*/return result;}

**strpbrk**

Funkcja zwraca adres pierwszego znaku z s1 należącego do zbioru set.

char\* strpbrk(const char \*s, const char \*set)

{char \*temp;while (\*s)

{temp = (char\*)set;

while (\*temp) /\* process set \*/

if (\*s == \*temp) return (char\*)s;

else temp++;s++;}return NULL;}

**strrchr**

Zwraca adres ostatniego wystąpienia znaku c, lub NULL gdy w łańcuchu s1 nie ma

tego znaku.

char \*strrchr(const char \*s, int c)

{char\* result = NULL;while (\*s)

{if (\*s == c)result = (char\*)s;s++;

}return result;}

**strspn**

Funkcja zwraca indeks pierwszego znaku z s1 nie należącego do zbioru znakówset.

size\_t strspn(const char \*s, const char \*set)

{size\_t pos = 0;char \*temp;while (\*s)

{temp = (char\*)set;while (\*temp)

if (\*s != \*temp) return pos;

else temp++;s++;pos++;}return pos;}

**strstr**

Funkcja zwraca adres początku łańcucha s2 występującego w s1.

char\* my\_strstr(char\* s, char\* s1)

{if ((\*s1 == '\0') || (\*s == '\0'))return NULL;

while (\*s){char\* temp1 = s;char\* temp2 = s1;

while (\*temp1 == \*temp2 && \*temp1 && \*temp2)temp1++, temp2++;if (\*temp2 == '\0')

return s;elseif (\*temp1 == '\0')return NULL;

s++;}}

**strtok**

Funkcja zwraca kolejne słowa (ang. tokens), oddzielone znakami z s2

char \*strtok(char \*s1, const char \*s2){

static char\* result;char\* temp;

if (s1) result=s1;else while (\*result++);

temp = (char\*)s2;while (\*temp){

if (\*result == \*temp){temp = (char\*)s2;

result++;continue;}temp++;}

s1 = result;while (\*s1){temp = (char\*)s2;

while ((\*s1 != \*temp) && (\*temp)) temp++;

if (\*temp == 0) s1++;else break;}

\*s1 = 0;return result;}

Można zgrabnie zapisać tę funkcję przy użyciu funkcji strcspn i strspn

char \* strDup (char const \*str)

{if (str == NULL) return NULL;

size\_t len = strlen(str) + 1;

char\* copy = new char[len];

if (copy != NULL){memcpy(copy, str, len); }

return copy;}char \* strdup(char \*s)

{char \*result = (char\*)malloc(strlen(s) + 1);

if (result == (char\*)0)return (char\*)0;

strcpy(result, s);return result;}

**MEM**

**memset**

{register unsigned char \*ptr = (unsigned char\*)dest;while (len-- > 0)\*ptr++ = val;

return dest;}

**memchr** (src\_void, c, length)

register CONST PTR src\_void;int c;

size\_t length;{CONST unsigned char \*src = (CONST unsigned char \*)src\_void;while (--length >= 0)

{if (\*src == c)return (PTR)src;src++;}return NULL;}

**memcmp** (str1, str2, count)

{register unsigned char \*s1 = (unsigned char\*)str1;

register unsigned char \*s2 = (unsigned char\*)str2;

while (count-- > 0){if (\*s1++ != \*s2++)

return s1[-1] < s2[-1] ? -1 : 1;}return 0;}

**memcpy**(out, in, length)

{bcopy(in, out, length);return out;}