Лекція 13. Функції роботи з рядками та текстовими файлами.

План

1. Тип char та ASCII код
2. Функції роботи з символами (ctype.h)
3. Робота з рядком через покажчики
4. Текстові файли
5. Розбиття на слова

Представлення символів і рядків в Сі

Символ тексту – це 1 байтний тип даних, що розуміється як ціле та символ. Інформаційна ємність 2^8-1 = 255

ASCII (*стандартний код для інформаційного обміну*) система кодів, у якій числа від 0 до 127 включно поставлені у відповідність літерам, цифрам і символам пунктуації. ASCII — це семи-бітний код.

char - ціле зі знаком в діапазоні -127 ... + 127
unsigned char - ціле від 0-255

Для роботи з кирилицею використовуйте unsigned char

ASCII – таблиця (Морозов с.322)

0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95
` 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	| 124	} 125	~ 126	⌂ 127
А 128	Б 129	В 130	Г 131	Д 132	Е 133	Ж 134	З 135	И 136	Й 137	К 138	Л 139	М 140	Н 141	О 142	П 143
Р 144	С 145	Т 146	У 147	Ф 148	Х 149	Ц 150	Ч 151	Ш 152	Щ 153	Ъ 154	Ы 155	Ь 156	Э 157	Ю 158	Я 159
а 160	б 161	в 162	г 163	д 164	е 165	ж 166	з 167	и 168	й 169	к 170	л 171	м 172	н 173	о 174	п 175

Приклади обробки символів з урахуванням їх кодування.

Отримати символ десяткової цифри з значення цілої змінної, що лежить в діапазоні 0..9:

```
int n; char c; c = n + '0';
```

Отримати значення цілої змінної з символу десяткової цифри:

```
if (c >= '0' && c <= '9') n = c - '0';
```

Перетворити маленьку латинську букву у велику:

```
if (c> = 'a' && c <= 'z') c = c - 'a' + 'A';
```

Символьні рядки. Строкова константа.

Рядок у сі - це масив типу char, останній елемент якого зберігає термінальний символ '\0'. Числове значення цього символу 0, тому можна говорити, що масив закінчується нулем.

```
char word[10];
word[0] = 'A';
word[1] = 'B';
word[2] = 'C';
word[3] = '\0';
//word[3] = 0; еквівалентно
printf("%s", word);
```

Для виведення використовувався ключ %s. При цьому рядок виводиться до першого термінального символу, бо функція printf не знає розмір масиву word

Строкова константа.

Строкова константа - послідовність символів, укладена в подвійні лапки. Строкова константа автоматично доповнюється символом '\0'.

```
char word[] = "ABC"; //строкова константа
char text[] = {'H','E','L','L','O','\0' };
printf("%s %d %d\n", word,
strlen(word),sizeof(word));
printf("%s %d %d\n", text,
strlen(text),sizeof(text));
```

Результат роботи програми:

```
ABC 3 4
HELLO 5 6
```

Функції вводу виводу символів/рядків

Функції вводу виводу одного символу:

```
char C;
C = getchar();      scanf("%c", &C);
putchar(C);         printf("%c", C);
```

Функції вводу виводу рядків символів:

```
char s1[25], s2[100];//виділення пам’ятті
scanf("%24s",s1); //scanf_s("%s",s1,24);
printf("%s", s1);
gets(s2); //gets_s(s2);
puts(s2);
```

Функції вводу виводу символів/рядків

Функції вводу виводу одного символу:

Ввід	Вивід
`C = getchar();`	`putchar(C);`
`scanf("%c", &C);`	`printf("%c", C);`

Функції вводу рядків символів:

	Ввід до пробілу	Ввід до Enter
	`scanf("%24s",s);`	`gets(s);`
VS	`scanf_s("%s",s, 24);`	`gets_s(s);`

Функції виводу рядків символів:

Вивід	
`printf("%s",s);`	`puts(s);`

Функції роботи з символами

#include <ctype.h>

Функція	Сенс
`isalpha`	перевіряє чи є символ літерою
`isalnum`	перевіряє чи є символ числом чи літерою
`isdigit`	перевіряє чи є символ числом
`islower`	перевіряє чи є символ малою літерою
`isupper`	перевіряє чи є символ великою літерою
`tolower`	перетворює літеру у нижній регістр
`toupper`	перетворює літеру у верхній регістр

*Ці функції працюють лише для англійських літер

Функції роботи з символами: приклади

Завдання. Дано рядок, знайти кількість великих літер та перетворити їх у маленькі літери

```
char s[100], *p;
int k=0;
gets(s);
p = s; //покажчик на масив
for(; *p; p++) //прохід по рядку
```

Без функцій

```
if (*p>='A' && *p<='Z') { k++; *p = *p - 'A' + 'a';}
```

З функціями

```
if (isupper(*p)) { k++; *p = tolower(*p);}
```

```
puts(s);
printf ("k=%d",k);
```

Деякі функції роботи з рядками

#include <string.h>

Функція	Сенс
`strlen(char *s)`	повертає довжину рядка
`strcpy(char *s1, char *s2)`	копіює з рядка s2 у рядок s1, s1 затирається
`strcat(char *s1, char *s2)`	об'єднує рядки s1 та s2, результат записує у s1
`strcmp(char *s1, char *s2)`	порівнює рядки, та повертає 0 якщо рядки рівні, додатнє число, якщо перший рядок передує у словниковому порядку, та від'ємне якщо передує другий
`sprintf(char *buffer, const char*format [, argument, ...])`	записує форматований рядок у текстовий буфер

Фунції роботи з рядками: приклади

Функції об’єднання рядків

```
char s1[80]="Hello ", s2[]="word!!!";
(s1,s2);
printf("s1:%s len=%d\n",s1, strlen(s1));
strcpy(s1,s2);
printf("s1:%s len=%d\n",s1, strlen(s1));
```

Результат:

```
s1:Hello word!!! len=13
s1:word!!! len=7
```

strcat – об’єднує s1 та s2 та результат записує у s1

strcpy – копіює (перезаписує) із s1 у i2

Фунції роботи з рядками: приклади

Функція формування форматованого рядка

```
char str[100], month[]="September";
int day =12;
sprintf(str, "Month %s day %d", month, day);
printf("%s",str);
```

Результат:

```
Month September day 12
```

Фунції роботи з рядками: приклади

Функція порівняння рядків

```
char p[]="parol", buf[20];
printf("VVedit parol: ");
scanf_s("%s",buf,19);
if (strcmp(p,buf)==0) printf("Virno\n");
else printf("Ne virno\n");
```

Результат:

```
VVedit parol: parol
Virno
```

Робота з динамічними рядками

Спосіб обробки рядків не залежить від того якими масивами вони представлені статичними чи динамічними: всі функції роботи з рядками для динамічних рядків працюють аналогічно.

```
char h1[]=" Yes ",h2[]=" No ",h3[]= " Or “, *p;
int n1=strlen(h1);//вимірюємо розмір
int n2=strlen(h2); //вимірюємо розмір
int n3=strlen(h3); //вимірюємо розмір
p = (char *)malloc (n1+n2+n3+1); //виділяємо пам’ять, +1 на ‘\0’
strcpy(p, h1); //копіємо в p h1
strcat(p, h2); //об’єднуємо  p з h2, результат у p
strcat(p, h3); //об’єднуємо  p з h2, результат у p
printf("%s len=%d\n",p, strlen(p));
free(p); //очищуємо пам’ять
```

Результат:

```
Yes  No  Or  len=13
```

Файли - послідовні потоки символів

- В мові Сі та Сі++ файл розглядається як *потік* (*stream*), що представляє собою послідовність байтів, що записуються чи зчитуються.
- За замовчуванням, файли, з якими працюють стандартні функції введення-виведення є послідовними **текстовими.**
- Робота з файлами проходить в **сеансовому** режимі.

Функції відкриття та закриття файлу

Функція	Дія	Параметри і результат
`FILE* fopen(char *name, char *mode)`	Відкрити файл	Результат `FILE*` - покажчик на описувач файлу або `NULL`
`int fclose(FILE *fd)`	Закрити файл	**`fd`** - ідентифікатор файлу (покажчик на описувач)

Інформація про потік заноситься в структуру `FILE`, яка визначена у файлі `stdio.h`. Файл відкривається за допомогою функції `fopen`, яка повертає покажчик на структуру типу `FILE*`.

Функція відкриття файлу встановлює зв'язок між відкритим файлом і елементом `fd` у програмі. Функція закриття файлу перериває цей зв'язок

Обмеження в роботі з текстовим файлом

Файл можна відкрити для використання тільки одного виду з перерахованих операцій:

- **Читання** тексту з існуючого файлу;
- **Запис** тексту у створений файл;
- **Додавання** тексту в існуючий файл.

Тому при зміні вмісту текстового файлу його можна або повністю переписати з вхідного в вихідний, або читати в пам'ять повністю і редагувати його там.

Значення аргументу mode функції fopen

“r”	відкриття файлу без дозволу на модифікацію, файл відкривається лише для читання.
“w”	створення нового файлу тільки для запису, якщо файл із вказаним ім’ям вже існує, то він перезапишеться.
“a”	відкриття файлу тільки для додавання інформації в кінець файлу, якщо файл не існує, він створюється.
“r+”	відкриття існуючого файлу для читання та запису.
“w+”	створення нового файлу для читання та запису, якщо файл із вказаним ім’ям вже існує, то він перезаписується.
“a+”	відкриває файл у режимі читання та запису для додавання нової інформації у кінець файлу; якщо файл не існує, він створюється.

Функції вводу/виводу у текстовий файл

Посимвольний в/в з файлу	Параметри і результат
int getc(FILE *fd)	Код символу або EOF
int putc(int ch, FILE *fd)	Код символа или EOF

Функції вводу/виводу у текстовий файл

```
FILE *f;
  char c;
  int i=0;
  char s[100];
  FILE *fd1=fopen("test.txt","r"); //Читання файлу
  FILE *fd2=fopen("res.txt","w"); // Створення файлу для запису
            if (fd1==NULL || fd2==NULL) return -1;
            while ((c=getc(fd1)) != EOF)//посимвольне читання з
файлу
                { printf("%c",c);
                if ((isalpha(c)) && isupper(c)) s[i++]=c; }// Запис
великих латинських літер у масив
            for(i--;i>=0;i--) putc(tolower(s[i]),fd2); // Запис
маленьких латинських літер у файл у зворотньому порядку
```

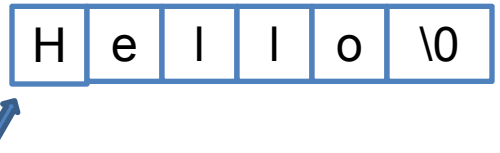
В наступному прикладі показується посимвольне копіювання вхідного файлу у вихідний.

Приклад роботи з символьними рядками через покажчики:

Цикл for виконується доти, доки *s=‘\0’, що є спеціальним символом кінця рядка.

```
int main()
{
  int num=0;
  char line[100]; //статичне виділення
пам’яті під масив із 100 символів
  gets(line); //ввід символів з клавіатури
  char *s; //змінна покажчик на char
  s=line; //s вказує на початок line
  for( ; *s; s++ ) //s проходить по line
     num++; //num рахує к-сть символіів
  printf("Kilkist simvoliv=%d", num);
  return 0;
}
```

line[]:



num = 8

На останньому кроці *s=‘\0’ , що є спеціальним символом кінця рядка.

Тому результат перевірки умови в for є true, що забезпечує вихід із циклу

Результат роботи програми:

```
hello
*s  = h, num= 1
*s  = e, num= 2
*s  = l, num= 3
*s  = l, num= 4
*s  = o, num= 5
Kilkist simvoliv=5
```

Тема: Приклади алгоритмів обробки рядків

Література:

1. Романов Е. Л. Практикум по програмированию.
   a) Розділ 4.4 Символи. Текст. Рядки С. 106-120
2. Вінник В. Ю. Основи програмування мовою Сі
   a) Розділ 8. Обробка текстових рядків. С. 104-118.

Всі фрагменти коду з цих розділів мають бути запушені у середовищі програмування та виконані!!!

Тема: Приклади алгоритмів обробки рядків

Приклад обробки математичного виразу

Завдання. Дано рядок, що зображає арифметичний вираз виду «<цифра> ± <цифра> ± ... ± <цифра>», де на місці знака операції «±» знаходиться символ «+» або «-» (наприклад, «4 + 7-2- 8 »). Вивести значення даного виразу (ціле число).

```
char *p, g[100],c;
int k=0;
   gets(g);
   p = g;//показчик приймає значення початку рядка
         //для проходу по константному масиву g
   for(;*p;p++)
   {
       if (*p == '+') {c='+';continue;} //запам'ятовування знаку +
       if (*p == '-') {c='-';continue;}//запам'ятовування знаку -
       if (*p >='0' && *p <='9')//дія, якщо цифра
       {
            if (c == '+') k+=*p -'0';
            else  if (c == '-')k-=*p -'0';
                else k=*p-'0';
       }
   }
   printf ("k=%d",k);
```

Тема: Приклади алгоритмів обробки рядків

Функції розбиття на слова та їх використання

Розбиття речення на слова. Потрібно врахувати, що програма не вміє просто «бачити слово», для неї необхідно формальна умова його виявлення. Таким може бути або кінець слова (не буква), або його початок (буква).

1. **Функція повертає покажчик на початок слова: char * strwordb (char* s)**
   a) **Отримує показчик на символу у рядку**
   b) **Перевіряє чи є він літерою (isalnum) , якшо так повертає показчик на нього**
   c) **Якщо ні шукає далі**
   d) **Якщо досягнуто кінець рядка повертається показчик вна нього**
2. **Функція повертає покажчик на кінець слова: char * strworde (char* s)**
   a) **Отримує показчик на символу у рядку**
   b) **Перевіряє чи є він (!isalnum) не літерою , якшо так повертає показчик на нього**
   c) **Якщо ні шукає далі**
   d) **Якщо досягнуто кінець рядка повертається показчик вна нього**
3. **Для використання цих функцій створєємо цикл, що ходить по рядку з виділенням слів: від початку слова шукаємо кінець, від кінця слова шукаємо початок наступного.**
4. **Ці функції можна використати для збереження слів у вільний масив (розділові знаки не збережуться)**

Тема: Приклади алгоритмів обробки рядків

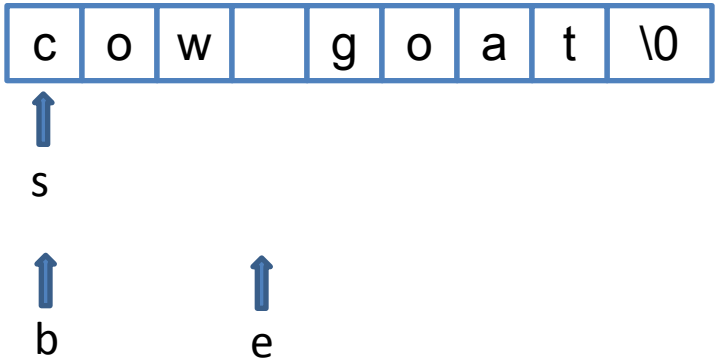
isalnum(int c) перевірка, чи є символ літерою або цифрою;

1 – повертає якщо символ є літерою або цифрою; 0 – у протилежному випадку

Функція повертає покажчик на початок слова

```
char *strwordb (char* s)
{  for( ; *s ; s++ ) //прохід по рядку
   if (isalnum(*s)) //якщо символ літера
        return s; //то повертаємо посилання
   return s;}
```

line[]:



Функція повертає покажчик на кінець слова

```
char *strworde (char* s)
{  for( ; *s ; s++ ) //прохід по рядку
    if (!isalnum(*s)) //якщо символ пробіл чи знак
        return s; //то повертаємо посилання
   return s;}
```

b – e = 4 max = 4

Функція повертає максимальну довжину слова

```
int MaxWord (char *s)
{ int max=0;
 char *b = strwordb(s),  char *e = NULL;
 for ( ;*b ;b = strwordb(e) ) {
    e=strworde(b);
    if( (e - b) > max) max = (e - b);}
return max;}
```

Знаходження найдовшого слова

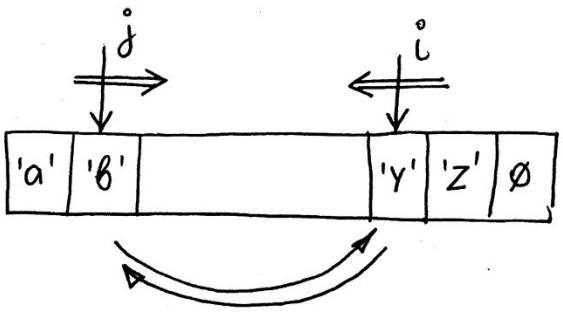
```
int  main()
{
 char line[] = "cow goat";
  printf("Najdovshe slovo = %d\n", MaxWord (line));
  return 0;
}
```

Тема: Приклади алгоритмів обробки рядків

Контекст: Рівномірний і нерівномірний рух в циклі

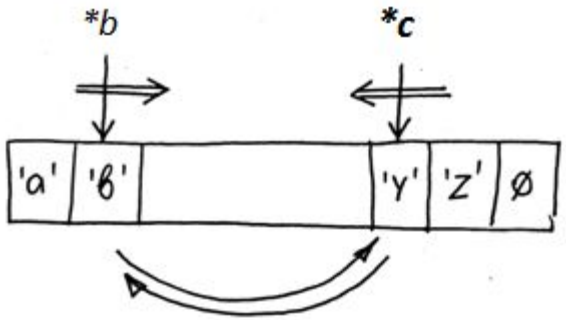
Приклад: *описати процедуру, що інвертує рядок*

```
void swap(char c[]){
int i,j;
// Цикл пошука кінца рядка для i
for (i=0; c[i] !='\0'; i++);
// Цикл попарного обміну
for (j=0,i--; i>j; i--,j++)
{ char s; s=c[i]; c[i]=c[j]; c[j]=s; }}
```



Те ж саме, але з показчиками

```
void swap1(char *c){
char *b;
b = c; //показчик на початок рядка
// Цикл переміщення показчика в кінець рядка
for (; *c; c++);
// Цикл попарного обміну
for (c--; c > b ; c--,b++)
{ char s; s=*c; *c=*b; *b=s; }}
```



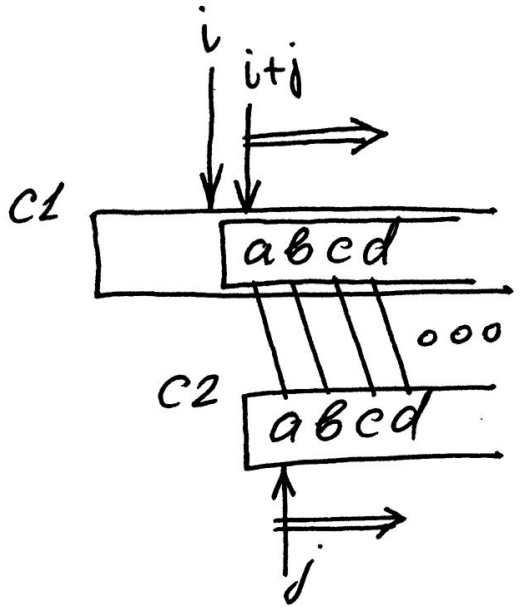
Показчики виконують роль індексів

Тема: Приклади алгоритмів обробки рядків

Контекст:Вкладені цикли і принцип відносності

При аналізі процесу, що проходить у внутрішньому циклі, зовнішній можна вважати «умовно нерухомим».

```
// --- Пошук підрядка в рядку
int search (char c1 [], char c2 [])
{ Int I, j;
  for (i = 0; c1 [i] != '\0'; i ++) {
       for (j = 0; c2 [j] != '\0'; j ++)
            if (c1 [i + j] != c2 [j]) break;
       if (c2 [j] == '\0') return i + 1;  }
return -1;}
```



Використання у main

```
char c1[100], c2[100];
printf ("Введіть перший рядок: ");
gets(c1);
printf ("Введіть підрядок, що будемо шукати: ");
gets(c2);
if (search(c1, c2)==-1) printf ("Підрядка не знайдено\n");
else printf ("Номер початку підрядка = %d\n", search(c1,
c2))
```

Зафіксувавши зовнішній цикл, c1 [i + j] слід розуміти як j-ий символ щодо поточного, на якому знаходиться зовнішній цикл. Звідси ми бачимо паралельний рух з попарним порівнянням символів за двома рядками, але другий розглядається від початку, а перший рядок, від i-го символу.