МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНА МЕТАЛУРГІЙНА АКАДЕМІЯ УКРАЇНИ

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з дисципліни

«АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ»
Частина 2
МОВА «С»

для студентів напряму 122 – «Комп’ютерні науки»

Дніпро - 2019

УДК 681.3.06+519.68

Методичні вказівки до виконання лабораторних робіт з дисципліні «Алгоритмізація та програмування». Частина 2. Мова “С”. Для студентів напряму і 122 «Комп’ютерні науки» Укл.: Н.Л. Дорош, О.О Кавац, Т.М. Фененко.– Дніпро: НМетАУ, 2019. – 25 с.

Методичні вказівки є першою частиною комплексу навчально-методичних матеріалів з дисципліні «Алгоритмізація та програмування»; містять теоретичні положення з мови програмування С, приклади програм, завдання з питань використання базових засобів мови С для самостійного виконання.

Призначені для студентів напряму 6.050101 00 – комп’ютерні науки, а також для слухачів курсів підвищення кваліфікації, студентів і аспірантів інших спеціальностей.

Укладачі: О.О Кавац, к.т.н., доц.
Н.Л. Дорош, к.т.н., доц.
Т.М. Фененко, ст. викл.

Відповідальний за випуск О.І. Михальов, д-р техн. наук, проф.

Рецензент О.І. Дерев’янко, канд. тех. наук., доц. (ДНУ)

Друкується за авторською редакцією.

Затверджено на засіданні НМК,
протокол № 5/14-15 від 25.05.2019

Підписано до друку 20.04.2019. Формат 60х84 1/16. Папір типогр. Друк різограф. Облік.-вид. арк 3,18. Умов. друк. арк. 3,14.
Тираж 100 пр. Замовл. № 30/2.

Національна металургійна академія України.
49600, Дніпро, пр. Гагаріна, 4

ЗМІСТ

Лабораторна робота №1 4
Функції. Робота з одновимірними масивами 4
Лабораторна робота №2 6
Функції. Робота з багатовимірними масивами 6
Лабораторна робота №3 9
Використання покажчиків на функції на прикладі використання функції QSORT 9
Лабораторна робота №4 13
Робота з рядками (масивами літер) 13
Лабораторна робота №5 15
Робота з файлами 15
Лабораторна робота №6 17
Робота з файлами, та рядками 17
ЛІТЕРАТУРА 19
ДОДАТОК 1 20

Лабораторна робота №1

ФУНКЦІЇ. РОБОТА З ОДНОВИМІРНИМИ МАСИВАМИ

Мета: Отримати навички процедурного програмування та навчитися використовувати одновимірні масиви як параметри функцій.

Завдання

Розробити програму, яка буде складатися з декількох функцій: головна функція, функція ініціалізації початкових даних, функція друку і функція обчислення. Головна функція повинна тільки визначати масиви та викликати відповідні функції обробки.

Програма повинна виводити початкову матрицю, результати обробки або модифіковану матрицю. Значення елементів матриць необхідно отримати, використовуючи генератор псевдо-випадкових чисел, згідно з умовами завдання.

Пам’ять для масивів треба розподіляти динамічно.

Номер завдання узяти згідно списку у журналі.

Теоретичні відомості

Нагадаємо деякі відомості щодо роботи з масивами:

Фрагмент програми, що динамічно виділяє пам’ять під масиви:

```
#include <stdlib.h>  // для calloc, malloc и др.
int  main(){
  double *m; // указатель на double – на начало массива
  int k; // число элементов массива
  . . .
  /*  Попытка выделения памяти с проверкой.Если неудача -выход
из программы  */
  if ((m = (double *) calloc(k,sizeof(double))) == NULL) {
    printf("Неудачная попытка выделения памяти\n");
    exit(1); }                  /* программа завершена */
  . . .
  free (m);  /* освобождение участка памяти, выделенного
calloc */
  return 0;
}
```

При передачі масивів до функцій, треба пам’ятати:

1. До функції масиви завжди передаються через адресу начала масиву;
2. Ім’я масиву уявляє собою константний покажчик на початок (на нульовий елемент) масиву, тобто наступні записи є еквівалентними: `mas == &mas == &mas[0]`
3. У списку формальних параметрів одновимірний масив можна описати так:
   `тип *имя` або `тип имя[]`, наприклад,
   `function1 (double *);` або `function1 (double []);`

Індивідуальні завдання до ЛР

1	Підрахувати кількість елементів масиву E, рівних нулю, індекси яких парні.
2	Обчислити добуток ненульових елементів масиву F, індекси яких непарні.
3	Обчислити суму елементів масиву M, значення яких перевищує індекс елемента.
4	Обчислити суму елементів масиву B , значення яких кратне п'яти і не перевищує задане число A.
5	Обчислити добуток елементів масиву S, значення яких не більше заданого

	негативного числа A, а індекси непарні.
6	Знайти максимальне число серед елементів масиву U, значення яких непарне.
7	Знайти номери елементів масиву S, значення яких парне.
8	Обчислити суму останніх десяти елементів масиву B.
9	Обчислити суму елементів масиву H, що не належать інтервалу [A, B].
10	Підрахувати кількість не додатних, додатних, нульових елементів масиву D.
11	Знайти середнє арифметичне елементів Q, значення яких менше A.
12	Знайти середнє геометричне елементів R, значення яких більше A.
13	Обчислити суму творів значень елементів масиву G на значення їх індексів.
14	Обчислити суму індексів елементів масиву W, у яких значення елементів більше A.
15	Обчислити суму всіх елементів масиву V, при обчисленні не додатні значення елементів замінювати їх абсолютним значенням.
16	Обчислити квадрат суми елементів, які більше заданого числа A і індекси яких кратні 3.
17	Вивести на друк елементи масиву Z, квадрат значення яких перевищує значення індексу в кубі.
18	Порівняти суму і добуток парних за індексом елементів масиву B, виключаючи нульові елементи.
19	Підрахувати кількість додатних елементів масиву X, значення яких в три рази більше індексу.
20	Обчислити добуток елементів масиву Y, значення яких не перевищує 3*i, де i – індекс елемента.
21	Обчислити суму елементів масиву C, значення яких парне і не перевищує задане число A.
22	Обчислити суму квадратів елементів масиву B, менших заданого числа A
23	Обчислити добуток елементів масиву S, значення яких не менше заданого позитивного числа A, а індекси кратні трьом.
24	Обчислити суму парних елементів масиву F, що перевищують задане число A.
25	Підрахувати кількість не додатних елементів масиву B, значення яких по модулю перевищує індекс елемента.
26	Знайти мінімальне число серед елементів масиву E, індекс яких кратний трьом.
27	Знайти номери елементів масиву D, значення яких непарне.
28	Обчислити добуток перших семи елементів масиву A.
29	Знайти номери елементів масиву M, рівних заданому числу A
30	Обчислити добуток елементів масиву C, що належать інтервалу [A,B].

Числові дані до першого індивідуального завдання

№	Тип елементів	Кількість елементів - N	Елементи		Додатково
			від	до	

1	Цілі	22	-4	4	
2	дійсні	15	-8	13	
3	Цілі	18	2	25	
4	Цілі	14	1	90	A=75
5	дійсні	16	-9	4	A=-4
6	Цілі	15	2	38	
7	Цілі	19	4	45	
8	дійсні	23	-14	42	
9	дійсні	17	-20	32	A=-4; B= 12
10	дійсні	25	-50	50	
11	дійсні	19	-15	21	A=11
12	дійсні	14	-12	10	A=5
13	дійсні	18	-8	14	
14	дійсні	20	-10	22	A=7
15	дійсні	17	-20	30	
16	дійсні	16	-5	15	A=9
17	дійсні	15	7	60	
18	дійсні	19	-13	11	
19	цілі	20	1	70	
20	дійсні	17	3	60	
21	цілі	15	2	18	A=10
22	дійсні	18	-10	21	A=5
23	дійсні	24	-18	12	A=3
24	цілі	12	2	25	A=8
25	дійсні	15	-23	20	
26	дійсні	21	-12	14	
27	цілі	17	3	30	
28	дійсні	17	1	7	
29	цілі	15	-10	10	A=5
30	дійсні	12	-4	15	A=0,05; B = 4,4

Лабораторна робота №2

ФУНКЦІЇ. РОБОТА З БАГАТОВИМІРНИМИ МАСИВАМИ

Мета: **Отримати навички процедурного програмування та навчитися використовувати багатовимірні масиви як параметри функцій.**

Завдання

Розбити програму, яка буде складатися з декількох функцій: головна функція, функція ініціалізації початкових даних, функція друку і функція обчислення. Головна функція повинна тільки визначати масиви та викликати відповідні функції обробки.

Програма повинна виводити початкову матрицю, результати обробки або модифіковану матрицю. Значення елементів матриць необхідно отримати, використовуючи генератор псевдо-випадкових чисел, згідно з умовами завдання.

Пам’ять для масивів треба розподіляти динамічно.

Номер завдання узяти згідно списку у журналі.

Теоретичні відомості

Нагадаємо деякі відомості щодо роботи з масивами:

Фрагмент програми, що динамічно виділяє пам’ять під двовимірний масив:

```
#include <stdlib.h>  // для calloc, malloc и др.
int  main(){
```

```
  int i;
  double  **dmas; /* dmas - указатель на указатель (указатель
на                массив). */
  int n,m; // число строк и столбцов матрицы
  . . .
/* выделяем память для массива указателей на double, по числу
строк*/
  if ((dmas=(double **)malloc(n*sizeof(double *))) == NULL) {
    printf("Неудачная попытка выделения памяти \n");
    exit(1); }                    /* программа завершена */
/* выделяем память для массивов-(строк матрицы) построчно и их
начальные адреса
   запоминаем в массиве указателей. */
  for (i=0; i<n;i++)
    if ((dmas[i]=(double *)calloc(m,sizeof(double))) == NULL)
{
      printf("Неудачная попытка выделения памяти для %2d – й
строки\n", i);
      exit(1); }                   /* программа завершена */
  . . .
/* освобождение памяти (в обратном порядке) */
  for (i=0; i<n;i++) free(dmas[i]);
  free(dmas);
  return 0;
}
```

У списку формальних параметрів двовимірний масив можна описати так:

`тип **имя` або `тип *имя[],` наприклад,

`function1 (double **);` або `function1 (* double[]);`

Індивідуальні завдання до ЛР

1	В заданій матриці A(N,M) знайти рядки, що не містять не доданих елементів.
2	Із заданої матриці A(N,M) видалити рядок і стовпець, в яких знаходиться перший елемент, рівний нулю. Отриману матрицю ущільнити. Елементи матриць проглядати зліва направо і зверху вниз.
3	Отримати нову матрицю S(N, M)шляхом складання всіх елементів завданої матриці C(N, M) з її найбільшим за модулем елементом.
4	Замінити мінімальний елемент головної діагоналі матриці D(N, N) сумою елементів першого рядка.
5	Знайти і надрукувати, скільки не додатних елементів містяться в кожному рядку матриці матриці F(N, M).
6	Замінити останній стовпець матриці H(N, N) мінімальним елементом головної діагоналі.
7	Розташувати елементи всієї матриці C(N, M) за спаданням (зверху вниз, зліва направо).
8	Обчислити суму рядків матриці Z(N, N) з нульовими елементами на побічній діагоналі.
9	Збільшити значення кожного елемента стовпця матриці G(N, M) на значення першого елемента відповідного стовпця.
10	Знайти і надрукувати значення і індекси мінімального елемента в кожному рядку матриці W(N, M).

11	Обчислити суму парних елементів в кожному стовпці матриці A(N, M)
12	Розташувати елементи кожного рядка матриці A(N, M) за спаданням.
13	Отримати нову матрицю F(N, M) шляхом складання всіх елементів завданої матриці B(N, M) з її найменшим за модулем елементом.
14	Поміняти місцями рядок матриці P(N, N), що містить максимальний елемент головної діагоналі, із стовпцем, що містить мінімальний елемент головної діагоналі.
15	Визначити, скільки рядків заданої матриці U(N,M) містять хоча б один елемент із діапазону [A, B].
16	Із заданої матриці H(N,M) видалити рядок, в якому знаходиться перший не додатний елемент. Елементи матриць проглядати зліва направо і зверху вниз
17	В заданій матриці G(N,M) знайти індекси першого елемента, що перевершує середнє арифметичне всіх елементів. Елементи матриць проглядати зліва направо і зверху вниз.
18	В заданій матриці T(N,M), яка містить тільки цілі числа, замінити перший не додатний елемент максимальним елементом матриці. Якщо не додатних елементів немає, то вивести відповідний текст. Елементи матриць проглядати зліва направо і зверху вниз.
19	Отримати нову матрицю C(N, M)шляхом віднімання всіх її елементів з максимального елемента матриці A(N, M).
20	Замінити всі елементи матриці Y(N, N), розташовані вище головної діагоналі, максимальним елементом, розташованим на головній діагоналі.
21	Знайти кількість додатних елементів в кожному стовпці матриці Z(N, M).
22	Замінити перший рядок матриці X(N, N) максимальним елементом головної діагоналі.
23	Обчислити число непарних елементів в кожному рядку матриці W(N, M)
24	Дана матриця A(N, M). Визначити кількість рядків матриці A, що містять хоча б одну нульову компоненту.
25	Обчислити суму стовпців матриці F(N, N) з позитивними елементами на головній діагоналі.
26	Розташувати елементи всієї матриці B(N, M) за збільшенням (зліва направо, зверху вниз).
27	Помножити значення кожного елемента рядка матриці D(N, M) на значення останнього елемента відповідного рядка.
28	Знайти і надрукувати значення і індекси максимального елемента в кожному стовпці матриці C(N, M).
29	Поміняти місцями рядок, що містить максимальний елемент, з рядком, що містить мінімальний елемент матриці E(N, M).
30	Розташувати елементи кожного стовпця матриці A(N, M) за збільшенням.

Числові дані до другого індивідуального завдання

№	Тип елементів	Розміри матриці		Елементи		Додатково
		N-рядки	M-стовпчики	від	до	
1	дійсні	8	5	-12	20	
2	цілі	6	7	-9	9	
3	цілі	6	4	-18	18	
4	цілі	5		-7	14	
5	цілі	6	7	-10	37	
6	цілі	6		-24	43	

7	дійсні	5	4	-54	61	
8	цілі	7		-4	7	
9	дійсні	7	5	-8	16	
10	дійсні	6	5	-13	36	
11	цілі	7	6	2	51	
12	цілі	5	7	-31	12	
13	цілі	7	6	-15	25	
14	цілі	5		-8	19	
15	дійсні	4	6	-19	22	A=5;B=12
16	цілі	7	6	-20	30	
17	цілі	6	4	-14	16	
18	цілі	7	5	-12	23	
19	цілі	6	5	-15	45	
20	цілі	5		-17	48	
21	дійсні	5	6	-14	26	
22	цілі	7		-9	37	
23	цілі	5	7	1	45	
24	цілі	5	4	-5	6	
25	цілі	6		-10	24	
26	цілі	7	4	-40	50	
27	дійсні	4	6	-9	12	
28	цілі	6	5	-20	18	
29	цілі	4	7	-12	30	
30	дійсні	5	6	-9	25	

Лабораторна робота №3

ВИКОРИСТАННЯ ПОКАЖЧИКІВ НА ФУНКЦІЇ НА ПРИКЛАДІ ВИКОРИСТАННЯ ФУНКЦІЇ QSORT

Мета: **Навчитися користуватися покажчиками на функції та використовувати вбудовані функції.**

Завдання

Виконати індивідуальне завдання. Описати структуру, ввести початкові данні та надрукувати їх на екрані, виконати сортування та вивести результати на екран.

Сортування провести, використовуючи вбудованну функцію `qsort()`:

1. по числовому полю;
2. по текстовому полю.

Номер завдання узяти згідно списку у журналі.

Теоретичні відомості

Для сортуванні рядків можна використовувати вбудовану функцію `int strcmp(const char *s1, const char*s2);` прототип функції знаходиться у файлі string.h

Опишемо вбудовану функцію швидкого сортування `qsort()`:

```
void qsort(void *base, size_t nelem,
     size_t width, int (*fcmp)(const void *, const void *));
```

Це прототип функції швидкого сортування (знаходиться у файлі `stdlib.h`). Функція `qsort()` сортує вміст таблиці (масиву) однотипних елементів, неодноразово викликаючи функцію порівняння, підготовлену користувачем.

Для виклику функції порівняння її адреса повинна замістити покажчик `fcmp`, специфікований як формальний параметр. При використовуванні `qsort()` програміст повинен підготувати таблицю сортованих елементів у вигляді одновимірного масиву фіксованої довжини і написати функцію, що дозволяє порівнювати два будь-які елементи сортованої таблиці. Зупинимося на параметрах функції `qsort()`:

`base` - покажчик на початок таблиці (масиву) сортованих елементів (адреса нульового елементу масиву)

`fcmp` - покажчик на функцію порівняння, який одержує як параметри два покажчики p1, p2 на елементи таблиці і повертає залежно від результату порівняння ціле число:

якщо *p1 < *p2, функція `fcmp ()` повертає не додатне ціле < 0;
якщо *p1 == *p2, `fcmp ()` повертає 0;
якщо *p1 > *p2, `fcmp ()` повертає додатне ціле > 0.

`nelem` - кількість елементів в таблиці, які будуть сортуватися (ціла величина, не більша за розмір масиву);

`width` - розмір елементу таблиці (ціла величина, що визначає в байтах розмір одного елементу масиву);

Приклад сортування за допомогою `qsort()` наведен у додатку 2, більш складні приклади наведені у лекційному курсі

Індивідуальні завдання до ЛР

1. Скласти програму, що обробляє наступні дані про працевлаштування: назва фірми; посада;заробітна плата, досвід роботи. Відсортувати за:
   - посадою;
   - досвідом роботи;
2. Скласти програму, що обробляє наступні дані про ДТП: прізвище інспектора, що зафіксував ДТП; прізвище порушника; місце події; сума штрафу. Відсортувати за:
   - прізвищем інспектора;
   - сумою штрафу.
3. Скласти програму, що обробляє наступні дані про наявність обчислювальної техніки: назва підрозділу, назва відділу, чисельність персоналу, кількість комп'ютерів. Відсортувати за:
   - назвою підрозділу;
   - кількістю комп'ютерів.
4. Скласти програму, що обробляє наступні дані про заробітну плату: П.І.Б робітника, відділ, нарахована заробітна плата, сума вирахувань з робітника. Відсортувати за:
   - прізвищем робітника;
   - нарахованную заробітною платою.
5. Скласти програму, що обробляє наступні дані про диск: назва файлу, диск, каталог розміщення, розмір файлу. Відсортувати за:
   - назвою файлу;
   - розміром файлу.
6. Скласти програму, що обробляє наступні дані про АТС: прізвище абоненту, назва АТС, тривалість дзвінка, місто, з яким з'єднують. Відсортувати за:
   - прізвищем абонента;
   - тривалістю дзвінка.

7. Скласти програму, що обробляє наступні дані про АТС: прізвище абоненту, назва АТС, тривалість дзвінка, тип дзвінка (міжміський/місцевий). Відсортувати за:
- назвою АТС;
- тривалістю дзвінка.

8. Скласти програму, що обробляє наступні дані про бригаду: П.І.Б робочого, назва підрозділу, стать, стаж роботи. Відсортувати за:
- назвою підрозділу;
- стажем роботи.

9. Скласти програму, що обробляє наступні дані про виробництво: П.І.Б робочого, назва підрозділу, стать, сімейний стан, вік. Відсортувати за:
- прізвищем робочого;
- віком.

10. Скласти програму, що обробляє наступні дані про деканат: прізвище викладача, назва предмету, назва групи, кількість не склавших МСК. Відсортувати за:
- прізвищем викладача;
- кількістю не склавши МСК.

11. Скласти програму, що обробляє наступні дані про: П.І.Б робочого, назва підрозділу, час запізнення. Відсортувати за:
- прізвищем робочого;
- часом запізнення.

12. Скласти програму, що обробляє наступні дані про чемпіонат: прізвище спортсмена, назва команди, кількість балів, кількість виступів. Відсортувати за:
- прізвищем спортсмена;
- кількістю балів.

13. Скласти програму, що обробляє наступні дані про літак: П.І.Б пасажиру, код рейсу, вага багажу, кількість речей. Відсортувати за:
- прізвищем пасажиру;
- кількістю речей.

14. Скласти програму, що обробляє наступні дані про літак: П.І.Б пасажиру, код рейсу, вага пасажиру; вага багажу. Відсортувати за:
- прізвищем пасажиру;
- вагою пасажиру

15. Скласти програму, що обробляє наступні дані про школу: П.І.Б учня, назва класу, оцінка. Відсортувати за:
- прізвищем учня;
- оцінкою.

16. Скласти програму, що обробляє наступні дані про ресторан: номер столу, назва блюда, ціна блюда, прізвище кухаря. Відсортувати за:
- назвою блюда;
- ціною блюда.

17. Скласти програму, що обробляє наступні дані про автомобілі: марка авто; колір авто; рік випуску; пробіг. Відсортувати за:
- маркою авто;
- пробігом авто.

18. Скласти програму, що обробляє наступні дані про автомобілі: марка авто; колір авто; рік випуску; місце реєстрації П.І.Б. Відсортувати за:
- маркою авто;
- роком випуску.

19. Скласти програму, що обробляє наступні дані про рекламне агенство: назва видання, назва фірми, що рекламується, вартість реклами. Відсортувати за:
- назвою видання;
- вартістю реклами.

20. Скласти програму, що обробляє наступні дані про пошту: індекс поштового відділення; назва видання; ціна видання; кількість підписаних видань. Відсортувати за:
- назвою видання;
- кількістю підписаних видань.

21. Скласти програму, що обробляє наступні дані про банки: П.І.Б клієнта, назва банку; назва вкладу; сума вкладу. Відсортувати за:
- прізвищем клієнта;
- сумою вкладу.

22. Скласти програму, що обробляє наступні дані про поїзди: шифр вагону(рядок символів); шифр составу; вага вагону; місце призначення. Відсортувати за:
- шифром вагону;
- вагу для кожного составу.

23. Скласти програму, що обробляє наступні дані про ремонтну бригаду: назва бригади; найменування ремонту; кількість запчастин, що потрібні; кількість запчастин, що є у наявності. Відсортувати за:
- найменуванням ремонту;
- кількістю запчастин.

24. Скласти програму, що обробляє наступні дані про робітника: П.І.Б робітника, код цеху, норма випуску за планом, кількість фактично виробленої продукції. Відсортувати за:
- прізвищем робітника;
- кількістю фактично виробленої продукції;

25. Скласти програму, що обробляє наступні дані про гуртожиток : П.І.Б. студенту, назва гуртожитку, номер кімнати. Відсортувати за:
- прізвищем студента;
- номером кімнати.

26. Скласти програму, що обробляє наступні дані про кафедру: назва кафедри, факультет, кількість викладачів, кількість лаборантів. Відсортувати за:
- назвою кафедри;
- кількістю викладачів.

27. Скласти програму, що обробляє наступні дані про робітника: П.І.Б.; шифр (табельний номер); відділ, заробітна плата; термін роботи. Відсортувати за:
- прізвищем робітника;
- заробітною платою.

28. Скласти програму, що обробляє наступні дані про склад: код складу; код деталі; назва деталі, загальна кількість деталей; кількість забракованих.. Відсортувати за:
- назвою деталі;
- загальною кількістю деталей.

29. Скласти програму, що обробляє наступні дані про склад: код товару; назва товару; ціна за одиницю товару; кількість. Відсортувати за:
- назвою товару;
- ціною за одиницю товару.

30. Скласти програму, що обробляє наступні дані про студентів: П.І.Б. студента, група, середній бал у сесію. Відсортувати за:
- прізвищем студента;

- середнім балом у сесію.

Лабораторна робота №4

РОБОТА З РЯДКАМИ (МАСИВАМИ ЛІТЕР)

Мета: **Навчитися обробляти символьні масиви**.

Завдання

Для заданої послідовності символів, треба виконати індивідуальні завдання 1 і 2 згідно свого варіанту.

Кожна з програм повинна виводити початкові дані масиву(ів), результати обробки або модифікований масив.

Кожна з програм повинна бути універсальною, тобто треба програмувати, наприклад, максимальні значення індексів масивів у вигляді змінних.

У роботі слід використати функції обробки рядків, що знаходяться у стандартних бібліотеках.

Деякі приклади обробки рядків наведені у додатку 3 і у лекційному курсі.

Теоретичні відомості

Наведемо функції, що застосовуються при роботі з рядками.

Функції вводу-виводу (зі стандартної бібліотеки), які можна використовувати для обробки рядків.

Введення рядків: `int getchar (void), char *gets (char *s), int scanf(const char *format, ...), int sscanf (char *s, const char * format, ...), char *fgets(char *s, int n, FILE *stream), char *fgets(char *s, int n, FILE *stream).`

Друк рядків: `int putchar (int c), int puts (const char *s), int sprintf (char *s, const char *format, ...), int fputs(const char *s, FILE *stream).`

Функції зі стандартної бібліотеки обробки рядків

Функції копіювання рядків: `char *strcpy (char *s1, const char *s2), char *strncpy (char *s1, const char *s2, size_t n).`

Об'єднання рядків: `char *strcat (char *s1, const char *s2), char *strncat (char *s1, const char *s2, size_t n).`

Функції порівняння рядків: `int strcmp (const char *s1, const char *s2), int strncmp (const char *s1, const char *s2, size_t n).`

Функції пошуку рядків (у рядках): `char *strchr (const char *s, int c), size t strcspn (const char *s1, const char *s2), size t strspn (const char *s1, const char *s2), char *strpbrk (const char *s1, const char *s2), char *strrchr (const char *s, int c), char *strstr (const char *s1, const char *s2).`

Функція розбиття рядка на лексеми: `char *strtok (char *s1, const char *s2).`

Функції для роботи з блоками пам'яті, які використовуються для обробки рядків: `void *memcpy (void *s1, const void *s2, size_t n), void *memccpy (void *s1, const void *s2, int c, size_t n), void *memmove (void *s1, const void *s2, size_t n), void *memcmp (const`

```
void *s1, const void *s2, size_t n), void *memicmp (const
void *s1, const void *s2, size_t n), void *memchr (const
void *s, int c, size_t n), void *memset (const void *s, int
c, size_t n).
```

Функція знаходження довжини рядка: `int strlen (const char *s).`

Також існують інші функції, які роблять перевірку та перетворення у рядках.

Індивідуальні завдання до ЛР

Завдання 1

Виконати наступні модифікації рядка літер згідно свого варіанту:

1. Надрукувати початковий рядок.
2. Надрукувати літеру, яка повинна бути видалена з рядка (вона знаходиться у рамочці) і новий рядок.
3. Замінити одну послідовність літер іншою послідовністю(дивись стовпчик 3, нагорі – послідовність, яку потрібно міняти, під нею – нова послідовність).
4. Розбити одержаний рядок на декілька, узявши літеру із 4-го стовпчика як розмежувач.

№	Рядок, що слід обробляти	3	4
1	`printf(""ПРОСМОТРЕТЬ МАССИВ");break;`	`printf cprintf`	`;`
2	`switch(l) {{case 1:gotoxy(2, 7);}`	`switch(l) switch(len )`	`:`
3	`if(q==0)printf(”Введите номер”);`	`q=0 q!=0`	`”`
4	`while(feof(f)) fread(&&ak[q],sizeof(struct sp),1,f);`	`feof(f) !feof(f)`	`,`
5	`if(((f=fopen(fname,"r"))!=NULL) fseek(f,0,SEEK_SET);`	`fname filename`	`,`
6	`if(ww=72){l--; if(l<1) l=4;}`	`w=72 w==72`	`;`
7	`else{{clrscr();goto(2,11);}`	`goto gotoxy`	`(`
8	`gotoxy(5,y);;printf("%5.3f",ak[].cost);`	`ak[] ak[i]`	`;`
9	`if(isdigit(ak[[q]kod_post)==0) goto m;`	`ak[q] ak[q].`	`)`
10	`scanf("%%3d", TanMass[cur].Vse); fflush(stdin);`	`TanMass &TanMass`	`;`
11	`switch(Nom)){case 1  Poisk1();  break;}`	`case 1 case 1:`	`;`
12	`printf("\n Нажмите любую клавишу")); getc();`	`getc() getch()`	`"`
13	`if (fset<<0){lsek(s,ofset,SEEK_END);}`	`lsek lseek`	`,`
14	`clrscr;q=0;if((f=fopen(""kar.psk","rb"))!=NULL)`	`clrscr clrscr()`	`;`
15	`ak=((spisok)malloc(sizeof(spisok)*100); menu(); free(ak);`	`=(spisok) =(spisok*)`	`*`

16	main(){{texcolor(colr); _setcursortype(0);proga();}	texcolor textcolor	;
17	fread(&ms[i],,sizeof(ms()),1,load);if(feof (load)) break;	ms() ms[0]	]
18	mas[i]==masx[j]; if (j=1) mas[1]=first;	J=1 j==1	=
19	if ((FileUk = open(Path, ""ab"))== NULL){return 1;}	open fopen	,
20	double i=0,, j=0,Nom; clrscr();	J=0, j=0; int	;
21	s1=open((argv[],O_WRONLY|O_CREAT|O_BINARY,0644);	argv[] argv[4]	|
22	float array[]={{16.375,-17.004,-43.353,-15.530};	array[] array[4]	,
23	for (i=0; i<10; i++) printf ("mas[d]=%f\t", i,mas[i]);	mas[d] mas[%d]	;
24	int pl1[[6], pl2[], k1 = 0, k2 = 0, num[3], c;	pl2[] pl2[5]	[
25	ch==getch();if (ch==80) lab+=1;	lab+=1 lab++	;
26	struct str{{ char *; float kg;}	char * char t[5]	]
27	int i;FILE in;in=fopen(("t.dat","wb");	FILE in FILE *in	;
28	do { ch=getch();; } while(ch=13)	ch=13 ch!=13	}
29	gotoxy(52,,y); y++; printf("%1.1f",s);	Y++ y+=1	;
30	S++;gotoxy(6,y); scanf("d",&a[s].tn);	"d" "%d"	)

Лабораторна робота №5

РОБОТА З ФАЙЛАМИ

Мета: Навчитися використати функції роботи з потоками.

Завдання

За основу узяти завдання для роботи №2, але номери завдань зсунути на 10, тобто 1-й за списком у журналі бере 11 номер, а 30-й - 10 номер. Розбити завдання на дві програми:

1. Перша з програм буде заповняти масив початковими даними і записувати їх у файл.
2. Друга – буде зчитувати дані із файлу та обробляти їх.

Теоретичні відомості

Наведемо деякі функції, які застосовуються для роботы з файлами.

Відкриття потоку (файлу):

```
FILE *fopen(const char *filename, const char *mode);
```

Зчитування одного символу з потоку: `int fgetc(FILE *stream);`

Запис одного символу у поток: `int fputc(int c, FILE *stream)`;

Форматне виведення до потоку:

```
int fprintf (FILE *stream, const char *format [, argument, ...]);
```

Форматне введення з потоку:

```
int fscanf (FILE *stream, const char *format [, address, ...]);
```

Зчитування рядка з потоку: `char *fgets(char *s, int n, FILE *stream)`

Запис рядка до потоку: `int fputs(const char *s, FILE *stream);`

Пересилання до потоку завдану кількість байт:

```
size_t fwrite(const void *ptr, size_t size, size_t n, FILE*stream);
```

Зчитування з потоку завдану кількість байт

```
size_t fread(void *ptr, size_t size, size_t n, FILE *stream);
```

Встановка покажчика у потоці на нову позицію:

```
int fseek(FILE *stream, long offset, int whence);
```

Установка покажчика на начало потока: `void rewind(FILE *stream);`

Повертання значення покажчика у потоці: `long ftell(FILE *stream);`

Аналіз досягнення кінця файлу: `int feof(FILE *stream).`

Закриття потоку: `int fclose (FILE *stream);`

Обробка помилок при роботі з потоками: `void perror (const char * s);`

Нагадаємо деякі засоби роботи з потоками(файлами)

Відкриття файлу, який пов'язний з потоком:

```
if ((fp = fopen("t.txt",  "w"))== NULL) {
  perror("ошибка при открытии файла t.txt \n");
  exit(0); }
```

Треба пам'ятати, що файл можна відкрити у різних режимах, таких як: запис, додавання, читання -"w", "a","r", а також у комбінованих режимах - "w+","a+","r+", крім того, додавання суфіксу "b" відкриває файл у бінарному режимі.

Аналіз досягнення кінця файлу:

```
а) while(1) {
      fgetc(stream);
      if(feof(cfPtr)) {/* если  данные исчерпаны – то выход */
break;}
      j++; }

б) while(1) {
     if (fread(     ) == NULL) break;   /* if( (c=fgetc()) ==
EOF) */
     j++; }
в)   while (!feof(stream)) ;
```

Приклад програми, що використовує форматний обмін даними з файлом приведений у додатку 4.

Лабораторна робота №6

РОБОТА З ФАЙЛАМИ, ТА РЯДКАМИ

Мета: **Навчитися обробляти символьні рядки, що знаходяться у файлах.**

Завдання

Треба розробити програму, що зчитує початкові дані з файлу, обробляє їх та виводить результати на екран або у файл.

Номер завдання узяти згідно списку у журналі.

Коментарі

Деякі теоретичні вадомості та практичні приклади до цієї роботи приведені у роботах 4-5.

Індивідуальні завдання до ЛР

1. Написати програму для пошуку однакових рядків у текстових файлах. Програма виводить номери рядків у три стовпчика: у перший виводяться рядки, які є тільки у першому файлі, у другий - рядки, які є тільки у другому файлі, у третій - однакові рядки у двох файлах.
2. Переформувати рядки файлу на задану користувачем довжину рядка. Якщо довжина рядка файлу перевищує задану, то виконати перенесення символів на новий рядок.
3. Порівняти два текстових файла. Рядки, що існують у обох файлах, треба вивести у новій файл, вказавши при цьому для кожного рядка його номери у файлах, які порівнюються.
4. Текст, що знаходиться у файлі і записаний у один рядок, треба переформувати. Символ $ - червоний рядок. Рядки повинні бути не більш, ніж 60 символів.
5. Текст, що сформований по рядках, треба підрівняти по правому краю. Для рівняння додавайте пропуски між словами.
6. Відкорегувати завданий текст. # -видалити наступне слово, @ - видалити наступне речення.
7. Видалити з тексту підряд записані однакові слова. Виправлений абзац – переформувати.
8. Поздоровлення. Для списку прізвищ, що завдані, треба підготувати поздоровлення до визначеного свята. Побажання виводити випадково з наперед завданого списку. Наприклад, щастя, здоров'я, успіхів на роботі.
9. Є список співробітників, які належать до різних громадських об'єднань (профком, спілка книголюбів, комітет молоді і т.п.). Треба надрукувати запрошення на чергове засідання вказаної організації. Завдається вид організації, місце збору та час.
10. Перевірити (і виправити якщо треба) текст, у якому знаходяться жіночі та чоловічі імена. Кожне ім'я повинне починатися з заголовної букви. Список імен завдається заздалегідь.
11. Перший файл містить текст, у якому є слова і словосполучення, що пропущені. Замість їх у тексті надрукований символ $. Треба замінити пропуски ($) відповідними словами і словосполученнями, що знаходяться у допоміжному файлі.
12. Перевірити (і виправити, якщо треба) щоб кожне речення у тексті починалося з заголовної букви і після кожної крапки був хоча б один пропуск.
13. У файлі-словнику знайти слова, які можуть бути одержані конкатенацією інших слів. Наприклад: "БАРСУК" = "БАР" + "СУК"

14. У файлі-словнику знайти слова, які містять однакові букви і різняться тільки порядком розташування. Наприклад, (КОМАР, КОРМА).
15. У одному файлі завдано список ключових слів. Треба вивести з іншого файлу речення, які містять хоча б одно з ключових слів.
16. У завданому файлі підрахувати кількість входження кожного з перших 10 різних слів.
17. У тексті треба перевірити (і виправити, якщо треба) правила написання (за російською мовою) букв И, А, У замість Ы, Я, Ю після Ж, Ч , Ш, Щ. (Будемо вважати, що файл не містить виключень: «жюри», «брошюра», «парашют»).
18. Кожен рядок тексту у файлі відцентрувати за допомогою додаткових пропусків, що додаються ліворуч чи праворуч від рядку.
19. Для досить великого слова, що задається , треба знайти у файлі-словнику усі слова, що можуть бути одержані з букв завданого слова.
20. З файлу-словника треба вибрати найдовше слово, букви якого не повторюються.
21. З файлу-словника треба вибрати російські слова, які можна надрукувати за допомогою символів англійського алфавіту, що співпадають за кресленням з російськими буквами (A, B, E, K, M, H, O, P, C, T, X – усього одинадцять). Наприклад: «СВЕТА», «РОВ» і т.і.
22. Для завданого номеру місяця треба вивести свята, що приходяться на цей місяць. Наприклад: 1 . 1 січня – Новий рік; 7 січня – Різдво. Данні про свята повинні бути розташовані у файлі.
23. Завдано файл з текстом. Надрукувати окличні речення(що закінчуються !).
24. Завдано файл з текстом. Надрукувати текст на екрані, обмінюючи місцями кожні два сусідні слова.
25. Завдано файл з текстом. Надрукувати на екрані тільки цитати, тобто фрагменти тексту, укладені у лапки (“це цитата”).
26. Завдано файл з текстом. Надрукувати на екрані тільки ті речення, які містять завдану кількість слів.
27. Завдано файл з текстом. Надрукувати на екрані текст, замінюючи цифри “0”, “1”,...,”9” на слова “нуль”, “один”,...,”дев’ять”.
28. Завдано файл з текстом. Надрукувати текст так, щоб усі речення були розташовані у зворотньому порядку.
29. Завдано файл з текстом. Надрукувати речення, яке містить найбільшу кількість знаків пунктуації.
30. Завдано файл з текстом. Надрукувати кожне речення і вказати, скільки разів зустрілося у ньому слово, яке ввели з клавіатури.

ЛІТЕРАТУРА

1. Б. Керниган, Д. Ритчи. Язык программирования Си.: Пер. с англ. - М.: Финансы и статистика, 1992.-272
2. Х. Дейтл, П. Дейтл. Как программировать на С(С++): Пер. с англ. – М.: ЗАО «Издательство БИНОМ», 2000. – 1008с.
3. Б. Страуструп. Язык программирования СИ++, спец изд./Пер. с англ. –М.; СПб.: «Издательство БИНОМ»-«Невский Диалект», 2001. –1099с., ил.
4. С.С++ Программирование на языке высокого уровня./Т.А. Павловская. – СПб:Питер, 2002. –464с;ил.
5. Подбельский В.В., Фомин С.С. Программирование на языке Си: Учеб. пособие.- 2-е доп. изд.- М.: Финансы и статистика, 2001.- 600 с.: ил.
6. Подбельский В.В. Язык Си++: Учеб. пособие.- 2-е изд., перераб. и доп.- М.: Финансы и статистика, 1996.- 560 с.: ил.
7. Стивен Прата. Язык программирования С++. Лекции и упражнения. Учебник: Пер. с англ./Стивен Прата – К.: Издательство «ДиаСофт», 2001. –656с.
8. Borland C++ 3.0. Programmer's Guide. – Scotts Valley, USA: Borland International, Inc. –1991. –467 p.
9. Borland C++ 3.0. User's Guide. – Scotts Valley, USA: Borland International, Inc. – 1991. –229 p.
10. Borland C++ 3.0. Library Reference. – Scotts Valley, USA: Borland International, Inc. –1991. –655 p.

ДОДАТОК 1

ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

1. Уважно прочитати і зрозуміти умову задачі, яку належить вирішити.
2. Ознайомитися з необхідним теоретичним матеріалом.
3. Вивчити приклади, додатки до відповідної лабораторної роботи, особливу увагу надати лістингу програми (для повного розуміння, можна виконати програму на комп'ютері).
4. Виконати загальні приклади.
5. Підготувати свій варіант програми і вирішити його за допомогою комп'ютера. При введенні і виведенні даних програма повинна видавати запрошення, коментарі і підказки. Програма повинна бути написана у загальному вигляді, тобто початкові дані, розміри масивів і т.п. треба завдавати змінними, значення яких можна перевизначити або ввести з клавіатури (файлу).
6. Зробити звіт.
7. Захистити виконану роботу, показавши знання теоретичного матеріалу по пройденій темі і продемонструвавши виконання завдання на комп'ютері.

Кожний звіт оформляється у вигляді записки пояснення згідно ГОСТ 19.001-77 — ЕДИНАЯ СИСТЕМА ПРОГРАММНОЙ ДОКУМЕНТАЦИИ і ГОСТ 2.105- 95 - ЕДИНАЯ СИСТЕМА КОНСТРУКТОРСКОЙ ДОКУМЕНТАЦИИ (на окремих листах формату А4 або в зошиті для лабораторних робіт) і повинен містити наступні елементи:

- титульний лист (при оформленні на окремих листах). При оформленні в зошиті необхідно вказати номер роботи, номер варіанту, прізвище і групу студента, що виконав роботу;
- текст записки пояснення в машинописному або рукописному вигляді;
- список використаної літератури;
- лістинг програми на мові Borland C: роздрукований на принтері або переписаний від руки і електронний варіант – файл з програмою.

Зміст записки пояснення

1. Постановка задачі.
2. Короткі теоретичні відомості про особливості вживання операторів і методів (теоретичне введення).
3. Опис програми:
   - загальні відомості (мова програмування, операційна система, тип процесора);
   - опис логічної структури програми;
   - опис алгоритму рішення задачі (у вигляді блок-схеми згідно ГОСТ 19.002-80 і ГОСТ 19.003-80);
   - опис початкових і вихідних даних програми;
   - опис функцій;
   - перелік аномалій і допустимих значень початкових даних (тестові приклади).
4. Висновки.

ДОДАТОК 2

Приклад сортування масиву рядків за допомогою `qsort()`, більш складні приклади наведені у лекційному курсі.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int sort_function( const void *a, const void *b);
char list[5][4] = { "cat", "car", "cab", "cap", "can" };
int main(void){
   int  x;
  printf("\n  До сортировки:",n);
  for (x = 0; x < n; x++)   printf("\npc [%d] = %p -> %s",
       x,list[x],list[x]);
   qsort((void *)list, 5, sizeof(list[0]), sort_function);
   printf("\n  После сортировкии:");
  for (x = 0; x < n; x++)   printf("\npc [%d] = %p -> %s",
   x,list[x],list[x]);
   return 0;
}
int sort_function( const void *a, const void *b) {
   return( strcmp((char *)a,(char *)b) ); }
```

Результати роботи програми:

```
  До сортировки:
pc [0] = 00AA -> cat
pc [1] = 00AE -> car
pc [2] = 00B2 -> cab
pc [3] = 00B6 -> cap
pc [4] = 00BA -> can
  После сортировкии:
pc [0] = 00AA -> cab
pc [1] = 00AE -> can
pc [2] = 00B2 -> cap
pc [3] = 00B6 -> car
pc [4] = 00BA -> cat
```

ДОДАТОК З

Приклад 1

Приклад розбиття рядка символів на окремі слова. Окремі слова будуть зберігатися у масиві slv. Це варіант з зафіксованою кількістю покажчиків(слів). Необхідно "доробити" програму так, щоб кількість покажчиків визначалася кількістю слів у фразі.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
int  main () {
  int i, n, dl, dlslv;
  char s[300]={'\0'};
  char *slv[30], *sn, *sk;
  //*slv[30] массив указателей для хранения слов
  // *sn, *sk - указатели на строку (на начало, конец
фрагмента)
  clrscr();
  puts("Введите предложение которое состоит из слов,
разделенных пробелами");
  puts("число слов от 1 до 30");
  fgets(s,300, stdin);
  /* Проверка - ввел пользователь строку? */
  if ((dl=strlen(s))==0) {puts("строка нулевой длины"); return
1;}
  // Разбиваем фразу на слова
  n=0; // число слов
  sn=&s[0];
  do {  // раз строка не нулевая - хотя бы одно слово есть
    n++;
    // Ищем пробел (разделитель). Если не найден то конец
слова - конец строки.
    if ( (sk=strchr(sn, ' ')) == NULL)
      { sk=&s[dl]; dlslv=(strlen(sn)-strlen(sk)+1);}
     else
      dlslv=(strlen(sn)-strlen(sk));
    //выделяем память для найденного слова
    slv[n-1]=(char *)calloc(dlslv+1,sizeof(char) ); // dlslv +
1 - для '\0'
    // копируем слово из фразы в массив указателей
    memcpy( slv[n-1], sn, dlslv);
    // и завершаем слово "нулем"
    slv[n-1][dlslv+1]='\0';
    printf("%d -е слово:  %s\n", n, slv[n-1]);
    sn=sk+1; // перемещаем указатель на следующее слово
    }      // начало нового слова за последним пробелом
      while(!(sk==&s[dl])); // достигли конец предложения
  printf("Всего %d слов\n",n);
while (!kbhit()); //ждем нажатия клавиш
```

```
return 0;}
```

Приклад 2

Підрахунок кількісті входжень кожного з символів до рядка.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
int main () {
  /* num - хранит число вхождений каждого символа в строку.
  индекс в массиве соответствует ASCII коду символа.  */
  int num[256], i, dl;
  unsigned char strok[255]={'\0'};
  clrscr();
  puts("Введите предложение");
  fgets(strok,255, stdin);
  /* Проверка - ввел пользователь строку? */
  if ((dl=strlen(strok))==0) {puts("строка нулевой длины");
return 1;}
  for( i=0; i<256; i++ )
      num[i] = 0;
  for( i=0; strok[i]; i++ )
      num[ (int) strok[i] ]++;
  // Все! подсчитали
  // Выводим начиная с пробела  (ASCII = 32 <=> 0x20)
    for( i=0x20; i<256; i++ ) {
      if( num[i] < 1 ) continue; // Нулевые вхождения символов
пропускаем
      printf( "%c - %d раз(а)\n", (unsigned char)i, num[i] );
    };
while (!kbhit()); //ждем нажатия клавиш
return 0;}
```

ДОДАТОК 4

Приклад форматного обміну даними з файлами. Програма створює файл int.dat и записує у нього символьне зображення чисел від 1 до 10 та їх квадратів.

```
#include <stdio.h>
int main() {
  FILE *fp; /* Указатель на поток */
  int n;
  if ((fp = fopen("int.dat","w")) == NULL) { perror("int.dat")
; return 1; }
  for    (n=1;    n<11;    n++)   fprintf(fp,    "%d      %d\n",
n,     n*n);
  fclose(fp);
  return  0;
}
```

Програма, форматного читання данних з файлу:

```
#include <stdio.h>
int main() {
  FILE *fp; /* Указатель на поток */
  int n, nn, i;
```

```
  if ((fp = fopen("int.dat","r")) == NULL) { perror("int.dat")
; return 1; }
  while (!feof(fp))
   { fscanf(fp,' "%d %d", &n,   &nn); printf(" %d %d \n",n,
nn); }
  fclose(fp);
  return 0;
}
```