

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

до виконання індивідуальних домашніх завдань

із дисципліни **"Програмування"**

для студентів спеціальності 123 "Комп'ютерна інженерія"
усіх форм навчання

2018

Методичні вказівки до виконання індивідуальних домашніх завдань із дисципліни "Програмування" для студентів спеціальності 123 "Комп'ютерна інженерія" усіх форм навчання / Укл.: Н.В. Луценко. – Запоріжжя: ЗНТУ, 2018. – 22 с.

Укладачі:

Н.В. Луценко, ст. викладач

Рецензент:

В.В.Шкарупило, к.т.н., доцент

Відповідальний за випуск

Н.В. Луценко, ст. викладач

Затверджено
на засіданні кафедри КСМ
Протокол № 9 від 17.05.2018

Рекомендовано до видання
на засіданні НМК факультету КНТ
Протокол № 10 від 28.05.2018

ЗМІСТ

Порядок виконання індивідуальних домашніх завдань.....	4
Розробка алгоритмів програм.....	4
Індивідуальні розрахункові завдання.....	5
Завдання № 1 Масиви динамічної пам'яті.....	5
Варіанти до завдання 1.....	7
Завдання № 2 Структури	9
Варіанти до завдання № 2.....	10
Завдання № 3 Функції і масиви.....	13
Варіанти до завдання № 3.....	14
Завдання № 4 Показчики на функції	17
Варіанти до завдання № 4.....	18
Рекомендована література	22

ПОРЯДОК ВИКОНАННЯ ІНДИВІДУАЛЬНИХ ДОМАШНІХ ЗАВДАНЬ

Індивідуальне домашнє завдання складається з декількох програм. Студент повинен розробити програми згідно заданого варіанту, налагодити і продемонструвати викладачу роботу розроблених програм.

Зміст індивідуального домашнього завдання

1. Титульний лист з номером варіанту
2. Тексти завдань згідно варіанту
3. Тексти програм з результатами виконання
4. Електронний носій з розробленими програмами

РОЗРОБКА АЛГОРИТМІВ ПРОГРАМ

Процес розробки програм можна розбити на такі етапи:

1) постановка завдання; 2) побудова математичної моделі та вибір методу рішення; 3) розробка алгоритму; 4) складання та налагодження програми; 5) обчислення і обробка результатів.

При постановці завдання головна увага приділяється з'ясуванню кінцевої мети, можливості вирішення досліджуваної проблеми, аналізу можливостей комп'ютері і даного середовища програмування.

При побудові математичної моделі визначається обсяг і специфіка вихідних даних, вводиться система умовних позначень, проводиться спрощення задачі. Вибір методу рішення включає в себе вибір чисельних методів (рішення систем рівнянь, диференціювання, інтегрування), визначення допустимої похибки при обчисленні і т.д.

При розробці алгоритму відбувається розкладання обчислювального процесу на складові частини, встановлюється порядок їх слідування, описується кожна частина (словесно або графічно).

Складання програми - розробка алгоритму на мові програмування (Сі, Паскаль, Фортран і т.д.). При налагодженні програми використовуються вбудовані отладчики, виявляються смислові і синтаксичні помилки, складаються тестові програми.

Після перевірки програми приступають безпосередньо до розрахунків за програмою, причому порядок роботи з програмою описується в інструкції користувача програмою.

ІНДИВІДУАЛЬНІ ДОМАШНІ ЗАВДАННЯ

Завдання № 1

Масиви динамічної пам'яті

Масиви динамічної пам'яті – це масиви, для даних яких виділяється динамічна пам'ять під час виконання програми в так званій "купі", розмір і положення якої залежать від моделі пам'яті. При цьому розмір масиву можна визначити як змінну цілого типу.

Для формування масивів динамічної пам'яті використовуються покажчики заданого типу та спеціальні функції і операції.

В мові Сі використовується функція **malloc()** для запиту і виділення потрібної кількості пам'яті в байтах. Ця функція повертає в програму адресу початку вільної ділянки пам'яті, тобто покажчик типу **void***, який потрібно перетворити до типу даних створюваного масиву. Для звільнення пам'яті служить функція **free()**. Прототипи цих функцій описані в заголовному файлі **<stdlib.h>**. Наприклад:

```
int n;
scanf("%d",&n); //введення розміру масиву (розмір – змінна)
int *ptri=(int*)malloc(n*sizeof(int)); //запит пам'яті
if(ptri==NULL) //перевірка виділення пам'яті
{puts("Error"); return -1;}
... //програма
free(ptri); // звільнення пам'яті
```

У мові С++ для цієї мети існують операції **new** і **delete**. Операція **new** автоматично визначає необхідний обсяг пам'яті в байтах та повертає покажчик заданого типу. Для звільнення пам'яті служить операція **delete[]**.

```
double *ptrd=new double[n]; //запит пам'яті
if(ptrd==NULL) //перевірка виділення пам'яті
{puts("Error"); return -1;}
... //програма
delete[] ptrd; // звільнення пам'яті
```

Звернення до елементів динамічного масиву може здійснюватися через індекси або за допомогою операції розіменування.

Наприклад:

```
for(i=0;i<n;i++)
    ptri[i]=10*i;
for(i=0;i<n;i++)
    printf("%lf ",*(ptrd+i));
```

Для створення двовимірного динамічного масиву визначається покажчик на масив покажчиків, для якого запитується пам'ять. Потім для кожного елементу масиву покажчиків теж запитується пам'ять

Синтаксис для створення двовимірного динамічного масиву за допомогою функцій **malloc** або операції **new** наступний:

```
//введення кількості елементів рядків n і елементів в рядку m
int n, m;
puts("Введіть розміри масиву");
scanf("%d %d", &n, &m);
int **ptr2; //визначення покажчика на масив покажчиків
//запит пам'яті для n покажчиків на рядки
ptr2=(int**)malloc(n*sizeof(int*));
// ptr2=new int*[n]; //запит пам'яті для n покажчиків на рядки
if(ptr2==NULL){puts("Помилка виділення пам'яті!");
               return -1;} // перевірка виділення пам'яті
//запит пам'яті для m елементів в кожному з n рядків
for(int i=0; i<n; i++)
{ptr2[i]=(int*)malloc(m*sizeof(int));
  // ptr2[i]=new int[m];
  if(ptr2[i]==NULL){puts("Помилка виділення пам'яті!");
                    free(ptr2); //звільнення пам'яті
                    // delete[] ptr2; //звільнення пам'яті
                    return -1;}
}
// програма
for(i=0; i<m; i++)
    free(ptr2[i]); //звільнення пам'яті
    // delete[] ptr2[i]; //звільнення пам'яті
free(ptr2); //звільнення пам'яті
// delete[] ptr2; //звільнення пам'яті
```

Доступ до елементів двовимірного динамічного масиву можна виконувати за допомогою двох індексів або двох операцій розіменування покажчиків на елементи. Наприклад:

```
for(i=0; i<n; i++)
    for(j=0; j<m; j++)
        ptri[i][j]=10*i+j;
for(i=0; i<n; i++)
    for(i=0; i<n; i++)
        printf("%d ", *((ptrd+i)+j));
```

Варіанти до завдання 1

Створіть двовимірний динамічний масив, розміри якого введіть з клавіатури. Виконайте обчислення згідно варіанту.

Варіант 1_1

Створіть динамічну квадратну матрицю елементів типу **float**, розмір якої введіть з клавіатури. Заповніть масив випадковими числами від -50.00 до 50.00. Обчисліть суму елементів першого рядка та другого стовпця. Виведіть на екран елементи масиву та обчислені суми. Наприкінці звільніть виділену під масив пам'ять.

Варіант 1_2

Створіть динамічний двовимірний масив елементів типу **char**, розміри якого введіть з клавіатури. Заповніть масив випадковими числами від 50 до 100. Обчисліть кількість парних елементів в другому рядку та в першому стовпці. Виведіть на екран елементи масиву та обчислені значення. Наприкінці звільніть виділену під масив пам'ять.

Варіант 1_3

Створіть динамічну квадратну матрицю елементів типу **long**, розмір якої введіть з клавіатури. Заповніть його випадковими числами від -5000 до 5000. Обчисліть кількість додатних елементів в другому рядку та в третьому стовпці. Виведіть на екран елементи масиву та обчислені значення. Наприкінці звільніть виділену під масив пам'ять.

Варіант 1_4

Створіть динамічний двовимірний масив елементів типу **int**, розміри якого введіть з клавіатури. Заповніть масив випадковими числами від 500 до 1000. Обчисліть кількість парних елементів та замініть парні елементи на 1. Виведіть на екран елементи масиву до і після зміни та обчислену кількість. Наприкінці звільніть виділену під масив пам'ять.

Варіант 1_5

Створіть динамічну квадратну матрицю елементів типу **float**, розмір якої введіть з клавіатури. Заповніть масив випадковими числами від +10.00 до +90.00. Обчисліть суму елементів третього рядка та першого стовпця. Виведіть на екран елементи масиву та обчислені суми. Наприкінці звільніть виділену під масив пам'ять.

Варіант 1_6

Створіть динамічний двовимірний масив елементів типу **int**, розміри якого введіть з клавіатури. Заповніть масив випадковими числами від -500 до +500. Обчисліть кількість від'ємних елементів та замініть від'ємні елементи на 0. Виведіть на екран елементи масиву до і після зміни та обчислену кількість. Наприкінці звільніть виділену під масив пам'ять.

Варіант 1_7

Створіть динамічну квадратну матрицю елементів типу **double**, розмір якої введіть з клавіатури. Заповніть масив випадковими числами від 0.000 до 30.000. Обчисліть суму елементів першого рядка та третього стовпця. Виведіть на екран елементи масиву до та після зміни. Наприкінці звільніть виділену під масив пам'ять.

Варіант 1_8

Створіть динамічний двовимірний масив елементів типу **long**, розміри якого введіть з клавіатури. Заповніть масив випадковими числами від 100 до 1000. Обчисліть кількість непарних елементів та замініть непарні елементи на 2. Виведіть на екран елементи масиву до і після зміни та обчислену кількість. Наприкінці звільніть виділену під масив пам'ять.

Варіант 1_9

Створіть динамічну квадратну матрицю елементів типу **int**, розмір якої введіть з клавіатури. Заповніть масив випадковими числами від 0 до 300. Замініть значення елементів двох діагоналей на 100. Виведіть на екран елементи масиву до та після зміни. Наприкінці звільніть виділену під масив пам'ять.

Варіант 1_10

Створіть динамічний двовимірний масив елементів типу **char**, розміри якого введіть з клавіатури. Заповніть масив випадковими числами від 50 до 100. Обчисліть кількість непарних елементів в другому рядку та в першому стовпці. Виведіть на екран елементи масиву та обчислені значення. Наприкінці звільніть виділену під масив пам'ять.

Завдання № 2

Структури

Структури – це сукупність поіменованих об'єктів у загальному випадку різних типів. Кожна структура включає в себе один або декілька об'єктів (змінних, масивів, покажчиків, структур), які називають елементами (полями) структури.

Структурний тип (*шаблон*) визначає, скільки елементів і якого типу входять в структуру. Визначення шаблону структури починається із слова **struct**, за яким у фігурних дужках розміщуються описи елементів, що входять в структуру. Шаблон структури має права типу. Під шаблон пам'ять не виділяється. Наприклад:

```
struct STR{ int a[2]; float r;};
```

Після визначення структурного типу (шаблону) за його допомогою можна визначити конкретні структури. Наприклад:

```
STR st1, st2[3], *ptrst=&st1;
```

Пам'ять для структур виділяється згідно структурного типу, тобто для структури **st1** виділиться **12** байт ($2 \cdot 4 + 4$), для масиву структур **st2** – **36** байт, для покажчика на структуру **ptrst** – **4** байти.

При визначенні структури можлива її ініціалізація, тобто привласнення початкових значень елементам структури. Для цього після визначення ставиться знак '=' та у фігурних дужках через кому задаються значення елементів згідно структурного типу. Наприклад:

```
STR st3={78,20,4.9};
```

```
STR st4[2]={ {23,20,4.4}, {15,19,4.3} };
```

Для звернення (доступу) до елементів структури використовуються уточнені імена. Наприклад:

```
st1.a[0]=19; st1.r=4.3;
```

Для звернення до елементів масиву структур використовується індекс, який вказують у квадратних дужках, наприклад:

```
st2[0].a[0]=20; st2[1].r=4.7;
```

При зверненні до елементів структури, зв'язаної з покажчиком, використовується операції стрілка або розіменування покажчика з ім'ям елемента структури після крапки. Нижче наведені обидві форми:

```
ptrst->a[1]=18; (*ptrst).r=4.5;
```

Можна створювати і знищувати структури з динамічною тривалістю існування. Для цього використовуються функції **malloc()** і **free()** або операції **new** і **delete**. Наприклад:

```

STR *st; // визначення покажчика на структуру типа STD
st=(STR*)malloc(sizeof(STR)); //Запит пам'яті
//st=new STR; //Запит пам'яті
if(st==NULL) // перевірка виділення пам'яті
{puts("Error"); return -1;}
scanf("%d %d %f",&st->a[0],&st->a[1],&st->r);
printf("%d %d %.2f\n",(*st).a[0],(*st).a[1],(*st).r);
free(st); // звільнення пам'яті
//delete st; // звільнення пам'яті

```

Варіанти до завдання № 2

Визначить заданий згідно варіанту структурний тип, за допомогою якого визначить структури, та виконайте обчислення.

Варіант 2_1

Визначить структурний тип, членами якого змінні типу **char**, **long** та масив з 3-х чисел типу **float**. За допомогою цього структурного типу визначить просту структуру, масив з 3-х структур та покажчик на структуру, під яку запросить динамічну пам'ять. Елементи перших двох структур з масиву проініціалізуйте, іншим елементам усіх структур привласніть будь-які значення. Визначить розмір пам'яті, що займають ці структури. Знайдіть найбільший член структури типу **long** серед усіх структур. Виведіть на екран значення структур та результат обчислення.

Варіант 2_2

Визначить структурний тип, членами якого є змінні типу **int**, **double** та масив з 3-х символів. За допомогою цього структурного типу визначить просту структуру, масив з 3-х структур та покажчик на структуру, під яку запросить динамічну пам'ять. Елементи перших двох структур з масиву проініціалізуйте, іншим елементам усіх структур привласніть будь-які значення. Визначить розмір пам'яті, що займають структури. Обчисліть суму членів типу **double** усіх цих структур. Виведіть на екран значення структур та результат обчислення.

Варіант 2_3

Визначить структурний тип, членами якого є символ, змінна типу **float** і масив із 3-х чисел типу **int**. За допомогою цього структурного типу визначить просту структуру, масив з 3-х структур та покажчик на структуру, під яку запросить динамічну пам'ять.

Елементи перших двох структур з масиву проініціалізуйте, іншим елементам усіх структур привласніть будь-які значення. Визначить розмір пам'яті, що займають ці структури. Знайдіть найменше число типу **int** серед усіх визначених структур. Виведіть на екран значення структур та результат обчислення.

Варіант 2_4

Визначить структурний тип, членами якого є змінні типу **int**, **char** та масив з 3-х змінних типу **float**. За допомогою цього структурного типу визначить просту структуру, масив з 3-х структур та покажчик на структуру, під яку запросить динамічну пам'ять. Елементи перших двох структур з масиву проініціалізуйте, іншим елементам структур привласніть будь-які значення. Визначить розмір пам'яті, що займають ці структури. Обчисліть добуток членів типу **float** усіх цих структур. Виведіть на екран значення структур та результат обчислення.

Варіант 2_5

Визначить структурний тип, членами якого є змінні типу **float**, **int** та рядок. За допомогою цього структурного типу визначить просту структуру, масив з 3-х структур та покажчик на структуру, під яку запросить динамічну пам'ять. Елементи перших двох структур з масиву проініціалізуйте, іншим елементам структур привласніть будь-які значення. Визначить розмір пам'яті, що займають ці структури. Обчисліть середнє значення членів типу **float** усіх цих структур. Виведіть на екран значення структур та результат обчислення.

Варіант 2_6

Визначить структурний тип, членами якого є рядок, змінні типу **long**, **float**. За допомогою цього структурного типу визначить просту структуру, масив з 3-х структур та покажчик на структуру, під яку запросить динамічну пам'ять. Елементи перших двох структур з масиву проініціалізуйте, іншим елементам структур привласніть будь-які значення. Визначить розмір пам'яті, що займають ці структури. Знайдіть найменший член типу **float** із усіх членів заданих структур. Виведіть на екран значення структур та результат обчислення.

Варіант 2_7

Визначить структурний тип, членами якого є масив із 3-х чисел типу **char** змінна типу **float** та масив з 2-х чисел типу **double**. За допомогою цього структурного типу визначить просту структуру,

масив з 3-х структур та покажчик на структуру, під яку запросить динамічну пам'ять. Елементи перших двох структур з масиву проініціалізуйте, іншим елементам структур привласніть будь-які значення. Визначить розмір пам'яті, що займають ці структури. Обчисліть добуток членів типу **double** усіх цих структур. Виведіть на екран значення структур та результат обчислення.

Варіант 2_8

Визначить структурний тип, членами якого є змінна типу **float**, масив з 4 змінних типу **long** та символ. За допомогою цього структурного типу визначить просту структуру, масив з 3-х структур та покажчик на структуру, під яку запросить динамічну пам'ять. Елементи перших двох структур з масиву проініціалізуйте, іншим елементам усіх структур привласніть будь-які значення. Визначить розмір пам'яті, що займають ці структури. Обчисліть суму членів типу **long** усіх цих структур. Виведіть на екран значення структур та результат обчислення.

Варіант 2_9

Визначить структурний тип, членами якого є символ і масив із трьох змінних типу **float** та змінна типу **long**. За допомогою цього структурного типу визначить просту структуру, масив з 3-х структур та покажчик на структуру, під яку запросить динамічну пам'ять. Елементи перших двох структур з масиву проініціалізуйте, іншим елементам структур привласніть будь-які значення. Визначить розмір пам'яті, що займають ці структури. Обчисліть серед усіх цих структур кількість додатних змінних типу **float**. Виведіть на екран значення структур та результат обчислення..

Варіант 2_10

Визначить структурний тип, членами якого є змінна типу **int**, рядок та масив із трьох змінних типу **float**. За допомогою цього структурного типу визначить просту структуру, масив з 3-х структур та покажчик на структуру, під яку запросить динамічну пам'ять. Елементи перших двох структур з масиву проініціалізуйте, іншим елементам усіх структур привласніть будь-які значення. Визначить розмір пам'яті, що займають ці структури. Обчисліть серед усіх цих структур кількість від'ємних змінних типу **float**. Виведіть на екран значення структур та результат обчислення.

Завдання № 3

Функції і масиви

Будь-яка функція, окрім головної функції **main()**, має бути визначена або описана до її використання (виклику).

Визначення функції має наступний синтаксис:

тип_функції ім'я_функції(сигнатура_функції) {тіло функції}

де **тип_функції** – це тип значення, що повертається функцією. Якщо функція нічого не повертає, то тип функції позначається **void**;

сигнатура_функції (список_формальних_параметрів)) – це список специфікацій окремих параметрів з вказівкою типів і імен, що записані через кому, або **void** чи порожньо, якщо у функцію не передаються ніякі параметри;

тіло функції – це послідовність описів, визначень змінних та різних операторів, що поміщені у фігурні дужки.

Звернення до функції (або *виклик функції*) – це вираз, в якому вказується ім'я функції та список фактичних параметрів в круглих дужках. Результат цього виразу – значення, що повертає функція:

ім'я_функції(список фактичних параметрів);

де **список фактичних параметрів** – це порожньо або **void**, (якщо в функцію нічого не передається), або фактичні параметри (константи або раніш визначені змінні, змінні з перетворенням типу, вирази).

При звернення до функції *формальні параметри* змінюються на *фактичні параметри* в строгій відповідності до типів параметрів по їх взаємному розташуванню в списках.

При передачі масиву у функцію як аргумент вказується ім'я масиву. Оскільки ім'я масиву - це покажчик-константа, то у функцію передається адреса початку масиву і усі дії усередині функції проводяться саме з елементами масиву оригіналу. При цьому список формальних параметрів в заголовку функції має дві рівноправні форми:

int func1(int arr[]){..}

чи **int func2(int *mass){..}**

У обох випадках у функції створюється копія покажчика відповідного типу, який діє усередині функції як звичайний покажчик-змінна і може змінювати своє значення. Доступ до елементів масиву усередині функції може здійснюватися за допомогою індексів, або через операцію розіменування покажчика.

Для роботи з усіма елементами масиву необхідно знати кількість елементів в масиві. При цьому в самій функції упізнати розмір масиву неможливо, оскільки у функцію передається покажчик на перший елемент масиву і розмір цього покажчика дорівнює 4 байти.

Є два способи визначення кінця масиву :1) одним параметром у функцію передається розмір масиву;2) останній елемент масиву задається яким-небудь унікальним значенням, яке аналізують у функції.

```
int sum(int x[],int N) // N визначає кількість елементів в масиві
{for(int S=0,i=0;i<N;i++)  S+=x[i];
  return S;}
void main()
{int a[5]={1,3,5,10,2},c[3]={2,1,6}, res1,res2;
  res1=sum(a,5); // виклик функції
  res1=sum(c,3); // виклик функції
  printf("res1=%d res2=%d\n", res1,res2);} // 21 9
```

Варіанти до завдання № 3

Розробіть задану згідно варіанту функцію та продемонструйте роботу цієї функції, зробивши в головній програмі три виклики функції з різними параметрами.

Варіант 3_1

Розробіть функцію, в яку передається масив типу **float**, що повертає індекс максимального додатного елемента цього масиву; або повертає **-1**, якщо додатних елементи в масиві немає. В головній програмі зробіть три виклики функції з різними масивами. При першому виклику елементи масиву проініціалізуйте від'ємними значеннями, при другому – задайте випадковими числами від **-5.00** до **+20.00**, при третьому – введіть з клавіатури. Всі масиви повинні мати різний розмір. Виведіть на екран елементи масиву та обчислені значення.

Варіант 3_2

Розробіть функцію, в яку передається масив типу **char**, що повертає індекс максимального парного елемента цього масиву; або повертає **-1**, якщо парні елементи в масиві відсутні. В головній програмі зробіть три виклики функції з різними масивами. При першому виклику елементи масиву проініціалізуйте непарними значеннями, при другому – задайте випадковими числами від **50** до **120**, при третьому – введіть з клавіатури. Всі масиви повинні мати різний розмір. Виведіть на екран елементи масиву та обчислені значення.

Варіант 3_3

Розробіть функцію, в яку передається масив типу **int**, що повертає індекс мінімального непарного елементу цього масиву; або повертає -1 , якщо непарні елементи в масиві відсутні. В головній програмі зробіть три виклики функції з різними масивами. При першому виклику елементи масиву проініціалізуйте парними значеннями, при другому – задайте випадковими числами від 60 до 140, при третьому – введіть з клавіатури. Всі масиви повинні мати різний розмір. Виведіть на екран елементи масиву та обчислені значення.

Варіант 3_4

Розробіть функцію, в яку передається масив типу **float**, що повертає індекс мінімального від'ємного елементу цього масиву; або повертає -1 , якщо додатні елементи в масиві відсутні. В головній програмі зробіть три виклики функції з різними масивами. При першому виклику елементи масиву проініціалізуйте додатними значеннями, при другому – задайте випадковими числами від -200.0 до 100.0 , при третьому – введіть з клавіатури. Всі масиви повинні мати різний розмір. Виведіть на екран елементи масиву та обчислені значення.

Варіант 3_5

Розробіть функцію, в яку передається масив типу **int**, що повертає добуток додатних парних елементів масиву, або повертає 0, якщо додатні парні елементи в масиві відсутні. В головній програмі зробіть три виклики функції з різними масивами. При першому виклику елементи масиву проініціалізуйте непарними значеннями, при другому – задайте випадковими числами від -6 до $+6$, при третьому – введіть з клавіатури. Всі масиви повинні мати різний розмір. Виведіть на екран елементи масиву та обчислені значення.

Варіант 3_6

Розробіть функцію, в яку передається масив типу **float**, що повертає індекс максимального від'ємного елементу цього масиву; або повертає -1 , якщо від'ємні елементи в масиві відсутні. В головній програмі зробіть три виклики функції з різними масивами. При першому виклику елементи масиву проініціалізуйте додатними значеннями, при другому – задайте випадковими числами від -150.0 до $+100.0$, при третьому – введіть з клавіатури. Всі масиви повинні мати різний розмір. Виведіть на екран елементи масиву та обчислені значення.

Варіант 3_7

Розробіть функцію, в яку передається масив типу **double**, що повертає середнє арифметичне додатних елементів масиву, менших **50.0** або повертає **0**, якщо такі елементи в масиві відсутні.. В головній програмі зробіть три виклики функції з різними масивами. При першому виклику елементи масиву проініціалізуйте від'ємними значеннями, при другому – задайте випадковими числами від **-50.0** до **700.0**, при третьому – введіть з клавіатури. Всі масиви повинні мати різний розмір. Виведіть на екран елементи масиву та обчислені значення.

Варіант 3_8

Розробіть функцію, в яку передається масив типу **float**, що повертає середнє арифметичне від'ємних елементів масиву від **-20.0** до **+20.00**, або повертає **0**, якщо такі елементи в масиві відсутні. В головній програмі зробіть три виклики функції з різними масивами. При першому виклику елементи масиву проініціалізуйте додатними значеннями, при другому – задайте випадковими числами від **-40.0** до **+100.0**, при третьому – введіть з клавіатури. Всі масиви повинні мати різний розмір. Виведіть на екран елементи масиву та обчислені значення.

Варіант 3_9

Розробіть функцію, в яку передається масив типу **long**, що повертає індекс максимального непарного елементу цього масиву; або повертає **-1**, якщо парні елементи в масиві відсутні. В головній програмі зробіть три виклики функції з різними масивами. При першому виклику елементи масиву проініціалізуйте непарними значеннями, при другому – задайте випадковими числами від **-100** до **+400**, при третьому – введіть з клавіатури. Всі масиви повинні мати різний розмір. Виведіть на екран елементи масиву та обчислені значення.

Варіант 3_10

Розробіть функцію, в яку передається масив типу **int**, що повертає добуток від'ємних непарних елементів масиву, або повертає **0**, якщо такі елементи в масиві відсутні. В головній програмі зробіть три виклики функції з різними масивами. При першому виклику елементи масиву проініціалізуйте парними значеннями, при другому – задайте випадковими числами від **-8** до **+8**, при третьому – введіть з клавіатури. Всі масиви повинні мати різний розмір. Виведіть на екран елементи масиву та обчислені значення.

Завдання № 4

Покажчики на функції

Покажчик на функцію визначається таким чином:

тип_функції(*ім'я_покажчика) (специфікація_параметрів);

де **тип_функції** - тип значення, що повертається функцією;

специфікація_параметрів - типи формальних параметрів;

***ім'я_покажчика** - ім'я покажчика на функцію в круглих дужках.

У визначенні покажчика на функцію тип значення, що повертається, і сигнатура функції повинні співпадати з відповідними типами і сигнатурами тих функцій, адреси яких передбачається присвоїти покажчику, що вводиться. Наприклад:

```
float mul(float x, float y) //визначення ф-ції mul
{return x*y;}
float(*ptrmul)(float,float); //визначення покажчика на функцію
int main()
{ptrmul=mul; // покажчику присвоїли ім'я функції mul
 float f1=3.0,f2=5.2,f3=0.5,f4=4.6,r1,r2;
  r1=(*ptrmul)(f1,f2); //виклик функції через покажчик на функцію
  printf("f1=%.2f f2=%.2f r1=%.2f ",f1,f2,r1);
// f1=3.00 f2=5.20 r1=15.60
  f2=(*ptrmul)(f3,f4); //виклик функції через покажчик на функцію
  printf("f3=%.2f f4=%.2f r2=%.2f ",f3,f4,r2);
// f3=0.50 f4=5.60 r2=2.30
  return 0;}
```

Для завдання типу покажчика на функцію можна застосовувати специфікатор **typedef**:

```
typedef int(*ABC)(int);
```

де **ABC** – ім'я типу покажчика на функцію, за допомогою якого можна описувати відповідні покажчики і їх масиви:

```
ABC p1, p2[2]; // покажчики на функцію типу ABC
```

Покажчики на функції можуть використовуватися як параметр функції. В цьому випадку у функцію можна передати покажчики на різні функції. Наприклад, функція **sum** підсумовує спочатку квадрати елементів масиву, а потім їх куби..

```
float sqr(float e) // визначення ф-ції sqr
{return e*e;}
float cube(float y) // визначення ф-ції cube
{return y*y*y;}
```

```
//визначення типу PF покажчика на функцію
typedef float>(*PF) (float) ;
// визначення ф-ції sum з використанням покажчика ф-цію fun типу PF
float sum(PF fun,float a[],int N) // визначення ф-ції sum
{float S=0.0;
  for(int i=0;i<N;i++)  S+=(*fun) (a[i]);
  return S;}
int main()
{float res1,res2,A[3]={1.5,2.0,3.0};
  res1=sum(sqr,A,3) ; //виклик функції з покажчиком на функцію
  res2=sum(cube,A,3) ; //виклик функції з покажчиком на функцію
  printf("res1=%.3f res2=%.3f\n",res1,res2) ;
  return 0;}
```

На екран виведеться:

```
res1=15.250 res2=38.375
```

Варіанти до завдання № 4

Розробіть програму з використанням покажчика на функцію, як параметра функції, згідно заданому варіанту. Продемонструйте роботу цих функцій.

Варіант 4_1

Визначте дві функції, що обчислюють максимальне (чи мінімальне) з трьох чисел типу **float**. Потім визначте функцію, що помножує суму заданих трьох чисел на максимальне (мінімальне) число, яке обчислюється за допомогою покажчика на з відповідну функцію. В головній програмі визначить і проініціалізуйте шість змінних. Зробіть чотири виклики функції. При першому виклику сума перших 3-х змінних помножується на максимальне число, при другому – на мінімальне, при третьому виклику сума останніх 3-х змінних помножується на максимальне число, при четвертому – на мінімальне число. Визначені змінні та результати обчислень виведіть на екран.

Варіант 4_2

Визначте дві функції, що обчислюють максимальний (чи мінімальний) елемент масиву типу **float** . Потім визначте функцію, що обчислює суму елементів масиву без обліку максимального (мінімального) числа, яке обчислюється за допомогою покажчика на відповідну функцію. В головній програмі визначить і проініціалізуйте два масиви різного розміру. Зробіть чотири виклики функції. При

першому виклику обчислюється сума елементів першого масиву без обліку максимального елемента цього масиву, при другому – без обліку мінімального елемента, при третьому виклику обчислюється сума елементів другого масиву без обліку максимального елемента цього масиву, при четвертому – без обліку мінімального елемента. Масиви та результати обчислень виведіть на екран.

Варіант 4_3

Визначте дві функції, що обчислюють суму (чи добуток) трьох чисел типу **float**. Потім визначте функцію, що помножає максимальне з трьох чисел на суму чи добуток цих чисел, які обчислюються за допомогою покажчика на відповідну функцію. В головній програмі визначить і проініціалізуйте шість змінних. Зробіть чотири виклики функції. При першому виклику максимальне число з перших 3-х змінних помножується на суму цих чисел, при другому – на добуток цих чисел, при третьому виклику мінімальне число з останніх 3-х змінних помножується на суму чисел цих чисел, при четвертому – на добуток цих чисел. Визначені змінні та результати обчислень виведіть на екран.

Варіант 4_4

Визначте дві функції, що обчислюють максимальне (чи мінімальне) з чотирьох чисел типу **float**. Потім визначте функцію, що обчислює суму заданих чотирьох чисел без обліку максимального (мінімального) числа, які обчислюються за допомогою покажчика на відповідну функцію. В головній програмі визначить і проініціалізуйте вісім змінних. Зробіть чотири виклики функції. При першому виклику обчислюється сума перших 4-х змінних без обліку максимального числа, при другому – без обліку мінімального числа, при третьому виклику обчислюється сума останніх 4-х змінних без обліку максимального числа, при четвертому – без обліку мінімального числа. Визначені змінні та результати обчислень виведіть на екран.

Варіант 4_5

Визначте дві функції, що обчислюють максимальний (чи мінімальний) елемент масиву типу **float**. Потім визначить функцію, що суму перших 3-х елементів масиву помножує на максимальний (мінімальний) елемент, який обчислюється за допомогою покажчика на відповідну функцію. В головній програмі визначить і проініціалізуйте два масиви різного розміру (розмір масивів повинен

бути більш 3). Зробіть чотири виклики функції. При першому виклику сума перших 3-х елементів першого масиву помножується на максимальний елемент цього масиву, при другому – на мінімальний елемент, при третьому виклику сума перших 3-х елементів другого масиву помножується на максимальний елемент цього масиву, при четвертому – на мінімальний елемент. Масиви та результати обчислень виведіть на екран.

Варіант 4_6

Визначте дві функції, що обчислюють суму (чи добуток) перших 3-х елементів масиву типу **float**. Потім визначте функцію, що ділить максимальний елемент масиву на суму чи добуток перших 3-х елементів масиву, які обчислюються за допомогою покажчика на задані функції. В головній програмі визначить і проініціалізує два масиви різного розміру. Зробіть чотири виклики функції. При першому виклику максимальний елемент першого масиву ділиться на суму перших 3-х елементів цього масиву, на при другому – на добуток перших 3-х елементів масиву, при третьому виклику максимальний елемент другого масиву ділиться на суму перших 3-х елементів цього масиву, при четвертому – на добуток перших 3-х елементів цього масиву. Масиви та результати обчислень виведіть на екран.

Варіант 4_7

Визначте дві функції, що обчислюють середнє арифметичне (чи середнє геометричне) трьох чисел типу **float**. Потім визначте функцію, що від суми трьох чисел відіймає середнє арифметичне (середнє геометричне), яке обчислюється за допомогою покажчика на відповідну функції. В головній програмі визначить і проініціалізує шість змінних. Зробіть чотири виклики функції. При першому виклику від суми перших трьох чисел відіймає середнє арифметичне цих чисел, при другому – середнє геометричне цих чисел, при третьому виклику від суми останніх трьох чисел відіймає середнє арифметичне цих чисел, при четвертому – середнє геометричне цих чисел. Визначені змінні та результати обчислень виведіть на екран.

Варіант 4_8

Визначте дві функції, що обчислюють суму додатних (від'ємних) елементів масиву типу **float**. Потім визначте функцію, що помножує максимальний елемент масиву на суму додатних (від'ємних) елементів масиву, яка обчислюється за допомогою

покажчика на відповідну функцію. В головній програмі визначить і проініціалізуйте два масиви різного розміру (розмір масивів повинен бути більш 3). Зробіть чотири виклики функції. При першому виклику максимальний елемент першого масиву помножується на суму додатних елементів цього масиву, при другому – на суму від’ємних елементів цього масиву, при третьому виклику максимальний елемент другого масиву помножується на суму додатних елементів цього масиву, при четвертому – на суму від’ємних елементів цього масиву. Масиви та результати обчислень виведіть на екран.

Варіант 4_9

Визначте дві функції, що обчислюють максимальний (чи мінімальний) елемент масиву типу **float**. Потім визначить функцію, що суму додатних елементів масиву помножує на максимальний (мінімальний) елемент, який обчислюється за допомогою покажчика на відповідну функцію. В головній програмі визначить і проініціалізуйте два масиви різного розміру. Зробіть чотири виклики функції. При першому виклику сума додатних елементів першого масиву помножується на максимальний елемент цього масиву, при другому – на мінімальний елемент цього масиву, при третьому сума додатних елементів другого масиву помножується на максимальний елемент цього масиву, при четвертому – на мінімальний елемент цього масиву. Масиви та результати обчислень виведіть на екран.

Варіант 4_10

Визначте дві функції, що обчислюють суму (чи добуток) останніх 3-х елементів масиву типу **float**. Потім визначте функцію, що помножує мінімальний елемент масиву на суму чи добуток останніх 3-х елементів масиву, які обчислюються за допомогою покажчика на задані функції. В головній програмі визначить і проініціалізуйте два масиви різного розміру (розмір масивів повинен бути більш 3). Зробіть чотири виклики функції. При першому виклику мінімальний елемент першого масиву помножується на суму останніх 3-х елементів цього масиву, при другому – на добуток останніх 3-х елементів цього масиву, при третьому виклику мінімальний елемент масиву другого масиву помножується на суму останніх 3-х елементів цього масиву, при четвертому – на добуток останніх 3-х елементів цього масиву. Масиви та результати обчислень виведіть на екран.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Подбельский В.В. Язык Си++: Учеб. пособие. – М.: Финансы и статистика, 2007. – 560 с.
2. Подбельский В.В. Практикум по программированию на языке Си: Учебн. пособие. – М.: Финансы и статистика, 2004. – 576 с.
3. Подбельский В.В., Фомин С.С. Программирование на языке Си: Учебн. пособие. – 2-е изд., – М.: Финансы и статистика, 2009. – 600с.
4. Павловская Т.А. С/С++. Программирование на языке высокого уровня. СПб: Питер, 2009.– 464 с.
5. Березин Б.И., Березин С.Б. Начальный курс С и С++. – М.: ДИАЛОГ-МИФИ, 2007. – 288 с.
6. Уильям Топп, Уильям Форд. Структуры данных в С++. / Пер. с англ.. – М.: ЗАО “Издательство БИНОМ”, 1999. – 816с.: ил.
7. Крячков А.В., Сухинина И.В., Томшин В.К. Программирование на С и С++. Практикум: Учебное пособие для вузов/ А.В.Крячков, И.В.Сухинина, В.К.Томшин.- 2-е изд.,исправ.-М.: Горячая линия.- Телеком, 2000.-344с.
8. С/С++. Структурное программирование. Практикум/ Т.А. Павловская, Ю.А.Щупак. – СПб.: Питер, 2002. – 240с.
9. Культин Н.Б. С/С++ в задачах и примерах.- СПб.: БХВ: Петербург, 2001. – 288с.
10. Х.М. Дейтел, П.Дж. Дейтел. Как программировать на С++. – 5-е изд. / Пер. с англ. – М.: ООО "Бином-Пресс", 2008.– 1456 с.
11. Шилдт, Герберт. Полный справочник по С++. – 4-е изд. / Пер. с англ. – М.: Вильямс, 2008. – 800 с.
12. Стивен Прата. Язык программирования С++. Лекции и упражнения. – 6-е изд. / Пер. с англ. – М.: Вильямс, 2012. – 478 с.