

1. Вказівник – це змінна, яка містить адресу іншої змінної, при чому вказівник вказує на змінну того типу, адресу якої він містить.

Вказівник оголошується за допомогою \*: <тип>\*<ім’я>;

1. . Вказівник, який не вказує на жодне значення, називається порожнім, чи нульовим. Такий вказівник має значення 0 чи NULL.
2. & – адресація, тобто отримання адреси змінної.

Наприклад, можна скористатись командою А = &х;

1. a) int a; double b; -- оголошення змінних цілих і дійсних типів.

б) int \*a; double \*b; -- оголошення вказівників на цілі та дійсні числа.

1. Пам’ять для звичайного масиву автоматично звільняється при виході з області видимості. Для звільнення пам’яті, виділеної під динамічний масив, використовується оператор delete[].Розмір звичайного масиву визначається під час компіляції і не може змінюватися під час виконання програми. Розмір динамічного масиву може бути встановлений під час виконання програми. Ви можете виділити пам’ять для динамічного масиву в залежності від потреби.
2. Правильні оголошення:

e) int \*a=(int\*) malloc(5\*sizeof(int));

f) int \*a=new int [5];

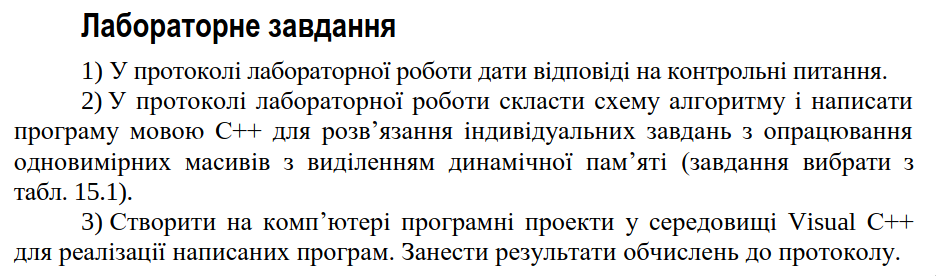
1. Пам'ять, виділену за допомогою оператора **new**, можна звільнити за допомогою оператора **delete**. Якщо пам'ять була виділена за допомогою функції **calloc**, її можна звільнити за допомогою функції **free**.
2. c) delete []a;

d) free (a);

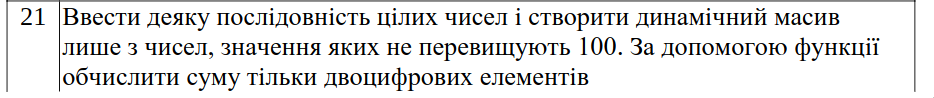
1. malloc -єдиний аргумент цієї функции size − кількість байтів, яку треба виділити. Функція повертає вказівник на початок виділеної пам’яті.

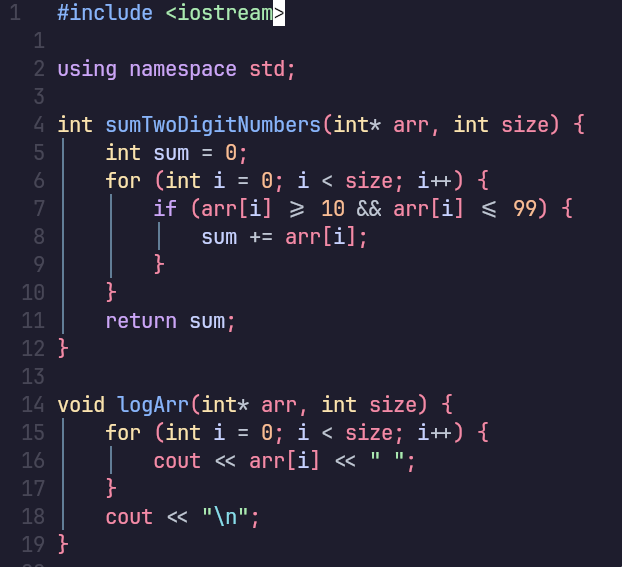
сalloc-виділяє блок пам’яті розміром numsize (під num елементів по size байтів кожен) і повертає вказівник на виділений блок. Кожен елемент виділеного блока ініціалізується нульовим значенням (на відміну від функції malloc).

1. Змінити (зменшити чи то збільшити) розмір виділеного раніш блока динамічної пам’яті можна функцією realloc, яка має такий формат: void \*realloc(\*bl, size);



**Таблиця 15.1**







#include <iostream>

using namespace std;

int sumTwoDigitNumbers(int\* arr, int size) {

int sum = 0;

for (int i = 0; i < size; i++) {

if (arr[i] >= 10 && arr[i] <= 99) {

sum += arr[i];

}

}

return sum;

}

void logArr(int\* arr, int size) {

for (int i = 0; i < size; i++) {

cout << arr[i] << " ";

}

cout << "\n";

}

int main() {

int n, element;

cout << "Введіть кількість чисел: ";

cin >> n;

int\* numbers = new int[n];

int validElementsCount = 0;

for (int i = 0; i < n; i++) {

cout << "Введіть елемент #" << i + 1 << ": ";

cin >> element;

if (element <= 100) {

numbers[validElementsCount] = element;

validElementsCount++;

}

}

cout << "Array before: \n";

logArr(numbers, n);

// Створення нового динамічного масиву для зберігання елементів, що не

// перевищують 100

int\* filteredNumbers = new int[validElementsCount];

for (int i = 0; i < validElementsCount; i++) {

filteredNumbers[i] = numbers[i];

}

cout << "Array after: \n";

logArr(filteredNumbers, validElementsCount);

cout << "Сума двоцифрових елементів: "

<< sumTwoDigitNumbers(numbers, validElementsCount) << endl;

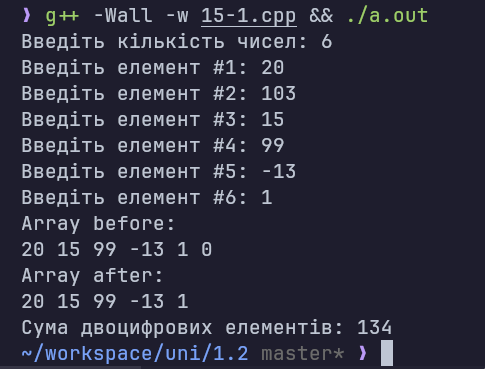
delete[] numbers;

delete[] filteredNumbers;

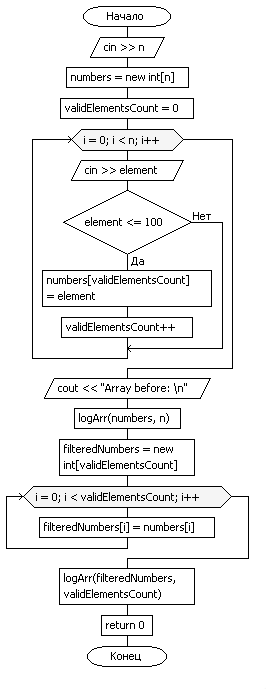
/\*delete[] newNumbers;\*/

return 0;

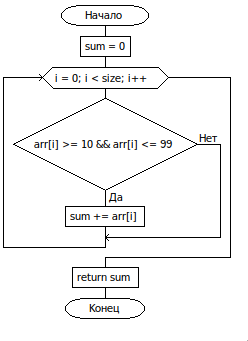
}



**main**



**sumTwoDigitNumbers**



**Висновки:** набув практичних навиків програмного використання вка-

зівників та динамічної пам’яті при опрацюванні одновимірних масивів.ы

**Роботу виконав: Холодир А. Р.**

**Роботу перевірив**