МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Миколаївський національний університет
імені В. О. Сухомлинського

О. С. Булгакова, В. В. Зосімов, Г. В.Ходякова

АЛГОРИТМІЗАЦІЯ І ПРОГРАМУВАННЯ: ТЕОРІЯ ТА ПРАКТИКА

Навчальний посібник

для дистанційного навчання

Миколаїв 2021

УДК 004.021
Б 90

РЕЦЕНЗЕНТИ:

Литвиненко В. І., Завідувач кафедри інформатики і комп’ютерних наук Херсонського національного технічного університету, Заслужений діяч науки і техніки України, доктор технічних наук, професор.

Атаманюк І. П., Завідувач кафедри вищої та прикладної математики Миколаївського національного аграрного університету, доктор технічних наук, професор.

Рекомендовано до друку вченою радою
Миколаївського національного університету імені В. О. Сухомлинського
(протокол № 27 від 29.06.2021 року)

Булгакова О. С.

Б 90 Алгоритмізація і програмування: теорія та практика : навчальний посібник для дистанційного навчання / О. С. Булгакова, В. В. Зосімов, Г. В. Ходякова. – Миколаїв: СПД Румянцева, 2021. – 138 с.

ISBN 867-345-684-657-5

Навчальний посібник є складовою частиною курсу «Алгоритмізація та програмування». В посібнику розглянуто базові алгоритми обробки даних, основи програмування мовою С++, правила написання коду, методи рішення задач. Всі головні теоретичні положення у посібнику супроводжуються прикладами вирішених задач. Передбачені матеріали для самостійної роботи студентів, тести для самоконтролю рівня засвоєння вивченого матеріалу, додаткова література для подальшого вивчення кожної теми. Посібник доповнює підручник або курс лекцій і поглиблює практичну складову курсу.

Навчальний посібник призначений для студентів механіко-математичного факультету Миколаївського національного університету імені В. О. Сухомлинського, які здобувають на першому рівні вищої освіти ступінь бакалавра за спеціальністю 122 Комп'ютерні науки. Крім того, посібник може використовуватися студентами інших суміжних спеціальностей.

УДК 004.021

ISBN 867-345-684-657-5

З М І С Т

ВСТУП

Даний навчальний посібник надає допомогу студентам першого курсу, які вивчають дисципліну "Алгоритмізація та програмування". Метою цього курсу є швидке засвоєння практичних умінь в програмуванні тих студентів, які мають різні вміння, досвід і рівень знань в цій області. Кожна нова тема в посібнику містить короткі теоретичні відомості та посилання на додаткові джерела. Практичні заняття передбачають паралельне вивчення базових алгоритмів та можливості мови С ++ в реалізації цих алгоритмів. Пропоновані завдання розбиті за категоріями. Частина запропонованих завдань містить зразки рішення. Їх слід виконувати в першу чергу. Потрібно набрати текст програми, скомпілювати, запустити на виконання, протестувати її роботу на різних наборах вхідних даних. На завдання з позначкою “алгоритм” слід звернути особливу увагу. Вони містять базові алгоритми, зміст яких, в тому числі словесне формулювання, потрібно вивчити. Багато задач, які пропонуються в посібнику для самостійної роботи, забезпечені підказками. Частину вирішуваних завдань студенти можуть перевіряти через сайт e-olymp.com. В кінці кожного розділу розміщені матеріали для самостійної роботи та тести для самоконтролю. В кінці посібника є термінологічний словник.

Навчальний посібник можна використовувати при проведенні лабораторних робіт, в самостійній роботі студентів, при оцінюванні знань і вмінь з дисципліни, що вивчається.

Розділ 1. Основні алгоритмічні структури.

1.1. Прості типи даних. Компілятори та середовища розробки на C++

Теоретичні відомості

Чим відрізняється мова програмування високого і низького рівня?

Мова програмування високого рівня написана мовою, зрозумілою людям. Найбільш поширеними є англомовні варіанти мов програмування. Наприклад**,** Pascal, C++, Java**,** Python та інші.

Мова низького рівня написана у формі інструкцій для комп'ютера. Також вона називається «машинною мовою», «в кодах» або «мовою асемблера».

Переклад програмного коду, написаного на мові високого рівня, в машинну мову для безпосереднього виконання називається компіляцією. А програма, яка виконує цей переклад, компілятором.

Інтегроване середовище розробки, ICP (англ. Integrated development environment — IDE) це комплекс програмних засобів, який використовується програмістами для написання програм та розробки програмного забезпечення.

Середовище розробки включає в себе:

- текстовий редактор,
- компілятор,
- засоби компонування, збірки або лінковки, які зв'язують воєдино всі об'єктні файли проекту,
- відладчик.

Для подальшої роботи нам будуть потрібні компілятор і середовище розробки. Для написання, налагодження і запуску програм досить

компілятора. Але для початку буде зручніше скористатися сервісними послугами інтегрованих середовищ розробки.

На домашньому комп'ютері потрібно установити компілятор мови С++ і, бажано, ще інтегроване середовище розробки.

Приклади компіляторів С ++

Gnu C ++

G ++ 4.9 C ++ 11

Microsoft Visual C ++

Середовища розробки (IDE):

Visual Studio

Code :: Blocks

NetBeans

CLion

Eclipse

Dev C ++

MinGV

C Free

Самостійна робота

Встановити на домашній комп'ютер компілятор Gnu C ++ і середовище розробки Code::Blocks.

Додаткові посилання

[1], [2], [3].

1.2. Найпростіша програма мовою С ++

Теоретичні відомості

Історія появи мови С ++ бере початок у 1972 р, коли Деннісом Рітчі і Брайаном Керніганом була розроблена мова програмування С, що поєднує в собі можливості мов високого і низького рівня, пізніше затверджена Американським Національним Інститутом Стандартизації (ANSI).

У 1980 р завдяки Б'ярну Страуструпу з'явився нащадок мови С – мова С++, що вібрав в себе позитивні риси ще кількох мов програмування. Мова С ++, на відміну від С, дозволяє програмісту розробляти програми з використанням як традиційного структурного, так і об'єктно-орієнтованого підходу.

Мова С ++, включає у себе ідентифікатори, ключові слова, функції, змінні, константи, оператори, вирази, директиви препроцесора, структури, масиви і ряд інших елементів. Більшість програм на С ++ починаються з оператора #include, який наказує компілятору включити зміст бібліотеки <iostream>, якщо iostream знаходиться в каталозі \ include. Бібліотека iostream пишеться в лапках #include "iostream ", якщо файл знаходиться в поточному каталозі. Бібліотека iostream містить оголошення функцій і змінних для потокового введення / виведення. Програма на С ++ обов'язково включає в себе функцію main(), з якої починається її виконання.

Розглянемо приклад найпростішої програми.

Завдання 1.1*. (приклад рішення)*

Ввести з клавіатури два цілих числа і вивести на екран їх суму.

```
#include <iostream>        // підключається бібліотека iostream
using namespace std;       // використовуємо стандартний простір імен
int main () {              // головний метод, точка запуску програми
        int a, b;          // оголошення двох змінних цілого типу
        cin >> a >> b;     // читаємо значення a і b за допомогою cin
        cout << a + b << endl;    // вивод суми на екран
        return 0;          // рекомендована команда
}                          // фігурні дужки містять тіло програми
```

cin >> x; – операція введення інформації із вхідного потоку (наприклад, з клавіатури). Функція cin може самостійно ідентифікувати, яка саме інформація введена: ціле число, дійсне число, символ або рядок;

endl переводить курсор на новий рядок;

Інструкція return забезпечує механізм завершення роботи функції. Функція main() повертає значення 0 за замовчуванням, якщо оператор return не використано явно, або повертає код помилки.

Самостійна робота

Зареєструватися на сайті e-olymp.com. Вивчити функціонал сайту.

Додаткові посилання

Сайт e-olymp.com. [Електронний ресурс]. – Режим доступу: https://www.e-olymp.com/ru/problems

1.3. Основні алгоритмічні структури та їх реалізація мовою С++

Теоретичні відомості

До основних алгоритмічних структур належать структури слідування, розгалуження і цикл.

Умовний оператор (розгалуження) має структуру:

if (умова) <оператор1>; else <оператор2>;

Оператор if виконує розгалуження програми в залежності від результату перевірки деякої умови на істинність:

Параметр умова може бути будь-яким виразом, але частіше за все воно містить оператори порівняння. Якщо перевірювана умова приймає істинне значення (true), виконується оператор1 . В іншому випадку (false) виконання програми переходить до оператору2.

У конструкціях мови С ++ оператори можуть бути блочними. Це означає, що в залежності від прийнятого рішення виконується не один, а цілий блок операторів. Блок операторів береться в фігурні дужки { } . Весь вміст блоку розглядається компілятором як єдиний оператор.

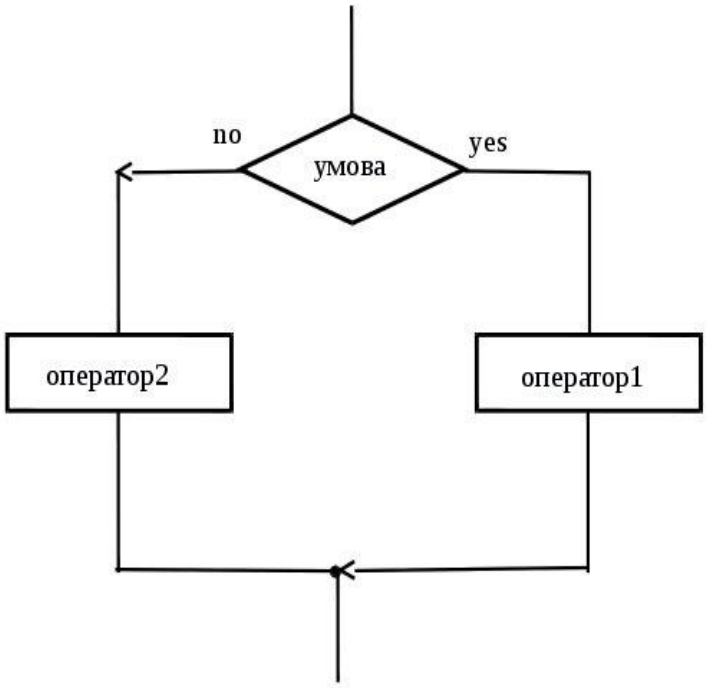


Рис.1.1. Блок-схема структури розгалуження

Зверніть увагу! При переміщенні по лініях інформаційного потоку зверху-вниз і зліва-направо стрілки на лініях ставити не обов'язково. Місце злиття потоків виділяється точкою.

Наступним потужним механізмом управління ходом послідовності виконання програми є використання циклів.

Цикл задає багаторазове проходження по одному і тому ж коду програми (ітерації). Він має точку входження, перевірочну умову і (необов'язково) точку виходу. Цикл, що не має точки виходу, називається нескінченним. Для нескінченного циклу перевірочна умова завжди приймає істинне значення.

Перевірка умови може здійснюватися перед виконанням тіла циклу (цикли for , while) або після його закінчення (do-while).

Цикли можуть бути з відомим числом повторень і з невідомим числом повторень.

Цикл for (з параметром)

```
for (int i = 0; i <100; i ++) {
        тіло циклу;
```

```
}
```

Круглі дужки після for включають в себе три секції:

- ініціалізація лічильника циклу: int i = 0;
- умова продовження циклу: i <100;
- зміна значення лічильника циклу: i ++;

Цикл while

```
i = 0;
while (умова) {
        тіло циклу;
        i ++;
}
```

Цикл do while

```
i = 0;
do {
        тіло циклу;
        i ++;
}
while (умова);
```

1.4. Приклади використання циклів

Теоретичні відомості

Як зробити затримку екрану – команда system("pause");

Завдання 1. 2. *(приклад рішення)*

Обчислити суму всіх цілих чисел від 1 до 1000.

1-й варіант рішення (за допомогою циклу while)

```
#include <iostream>
using namespace std;
```

```
int main () {
        int sum = 0, i = 1 ;
        while (i <= 1000) {
                sum += i;
                i ++;
        }
        cout << sum << endl;
        system("pause");
}
```

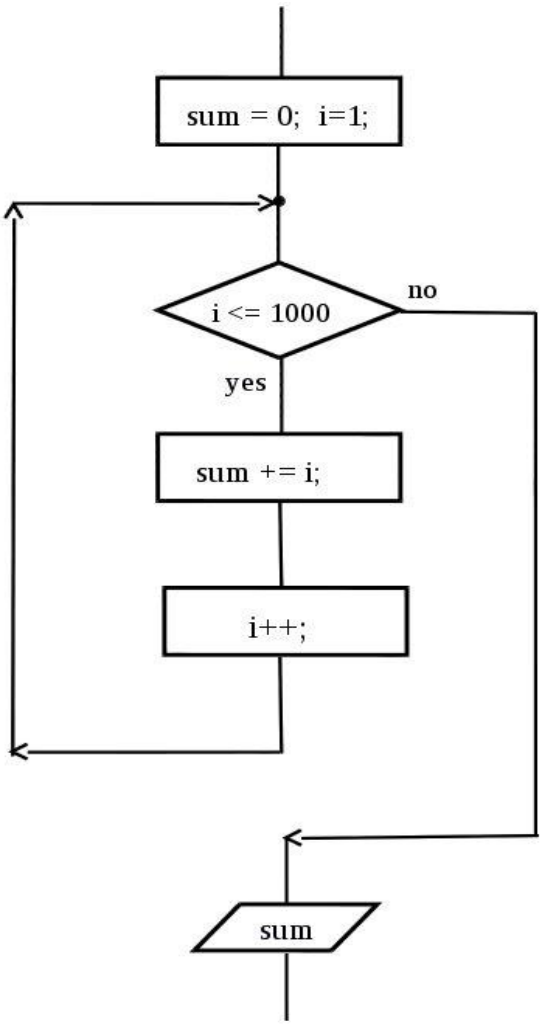


Рис.1.2. Блок-схема структури повторення

2-й варіант рішення (за допомогою циклу for)

```
#include <iostream>                    // cin, cout
```

```
using namespace std;
int main () {
        int sum = 0;
        for (int i = 1; i < = 1000; i ++) {
                sum = sum + i;
        }
        cout << sum << endl;
}
```

Завдання 1.3. *(самостійна робота)*

Знайти суму цілих чисел на проміжку від **a** до **b**

Задача 2860 на e-olymp.com.

Вхідні дані

Два цілих числа **a** і **b** , по модулю не перевищують 2 000 000 000.

Вихідні дані

Сума цілих чисел на проміжку від **a** до **b** .

Тестовий приклад

Вхідні дані	Вихідні дані
2 5	14

Підказка.

Використовуйте формулу суми членів арифметичної прогресії.

Завдання 1.4. *(приклад рішення)*

Дано дійсне число x і натуральне число n. Знайти суму членів ряду

$$1+\frac{x^1}{1}+\frac{x^2}{2}+\frac{x^3}{3}+\ldots+\frac{x^n}{n}$$

```
#include <iostream>
using namespace std;
int main () {
    double x;
    int n;
    cin >> x >> n;
    double sum = 1, num = 1;
    for (int i = 1; i <= n; i ++ ) {
       num * = x;
       sum + = num / i;
    }
    cout << sum << endl;
}
```

1.5. Алгоритми обміну значеннями двох змінних, пошуку найбільшого з двох і найбільшого з трьох чисел.

У цьому розділі розглянута реалізація базових алгоритмів, в тому числі, з використанням функцій стандартної бібліотеки C ++

Завдання 1.5. *(алгоритм)*

Обмін значеннями двох змінних a і b.

1-й варіант рішення

```
#include <iostream>
using namespace std;
int main() {
        int a, b, c;
        cin >> a >> b;
        c = a;  a = b;  b = c;
        cout << a << ' ' << b << endl;
}
```

2-й варіант рішення

```
#include <iostream>
using namespace std;
int main() {
        int a, b;
        cin >> a >> b;
        a = a + b;
        b = a – b;
        a = a – b;
        cout << a << ' ' << b << endl;
}
```

3-й варіант рішення (з використанням функції swap)

```
#include <iostream>
using namespace std;
int main () {
        int a, b;
        cin >> a >> b;
        swap(a,b);              // функція стандартної бібліотеки
        cout << a << ' ' << b << endl;
}
```

Завдання 1.6. *(алгоритм)*

Пошук найбільшого з двох чисел

1-й варіант рішення

```
#include <iostream>
using namespace std;
int main() {
        int a, b;
        cin >> a >> b;
        if (a> b)
                cout << a << endl;
```

```
        else
                cout << b << endl;
}
```

2-й варіант рішення (з використанням функції max)

```
#include <iostream>                // cin, cout
#include <cmath>                   // max
using namespace std;
int main() {
        int a, b;
        cin >> a >> b;
        cout << max(a, b) << endl;       // функція стандартної бібліотеки
}
```

3-й варіант рішення (з використанням тернарного оператора)

```
int max (int a, int b) {
        return a > b? a: b;
}
// тернарний оператор працює з трьома операндами,
c? x: y
```

де c - логічна змінна.

Якщо c == true, то повертається значення x, інакше – значення y.

Завдання 1.7. *(алгоритм)*

Пошук найбільшого з трьох чисел

1-й варіант рішення

```
#include <iostream>
using namespace std;
int main() {
        int a, b, c;
        cin >> a >> b >> c;
        int max_el;
```

```
        if (a> b)  max_el = a;
        else  max_el = b;
        if (c> max_el) max_el = c;
        cout << max_el << endl;
}
```

2-й варіант рішення (з використанням функції max)

```
#include <iostream>
#include <cmath>                    //max
using namespace std;
int main() {
        int a, b, c;
        cin >> a >> b >> c;
        cout << max (c, max (a, b)) << endl;
}
```

1.6. Змінні і основні типи даних в С ++

Теоретичні відомості

Змінна - це іменована область оперативної пам'яті, в якій зберігаються дані. У змінної є ім'я, тип, значення, час життя, область дії і область видимості.

Тип даних визначає розмір пам'яті, виділеної для зберігання змінної, набір операцій, які визначені для цього типу даних і діапазон значень.

Тип	Розмір пам'яті (в байтах)	Діапазон значень	Кількість знаків	Назва типу
bool	1	true, false	1	логічний тип даних
char	1	0 .. 255	1	символьний тип даних
int	4	-2 147 483 648 .. 2 147 483 647	10	цілочисельний
unsigned int	4	0 .. 4 294 967 295	10	беззнаковий int

long long	8	- 9223372036854775808 .. 9223372036854775807	19	цілочисельний
unsigned long long	8	0 .. 18446744073709551615	20	беззнаковий long long
float	4	від 1.2e-38 до 3.4e38	39	дійсний тип одинарної точності
double	8	від 2.2e-308 до 1.8e308	309	дійсний тип подвійної точності
// float і double – в позитивному діапазоні				

Таблиця 1.1. Основні прості типи даних в С++

Завдання 1.8. *(приклад)*

Отримання інформації про розмір пам'яті (в байтах), що виділяється під змінну даного типу:

```
cout << sizeof(int) << endl;                // 4
cout << sizeof(long long) << endl;          // 8
cout << sizeof(unsigned long long) << endl; // 8
```

Завдання 1.9. *(приклад)*

Отримання інформації про максимальне і мінімальне значення змінної типу int.

```
cout << INT_MAX << endl;    // 2 147 483 647
cout << INT_MIN << endl;    // – 2 147 483 648
```

Назва операції	Знак операції
складання	+
віднімання	–
множення	*
цілочисельне ділення	/
знаходження залишку від ділення	%

Таблиця 1.2. Операції, які визначені на множині int.

Приклади використання цілочисельного ділення і знаходження залишку від ділення:

23/10 = 2;

23% 10 = 3;

Для ефективного використання повернення значень операціями використовується модифікований оператор присвоювання.

Назва операції	Знак операції
складання з присвоюванням	+=
віднімання з присвоюванням	–=
множення з присвоюванням	*=
ділення з присвоюванням	/=
модуль з присвоюванням	%=

Таблиця 1.3. Модифікації оператора присвоювання

Завдання 1.10. *(самостійна робота)*

Програма зчитує двозначне число і виводить через пробіл кожну цифру окремо

Задача 1, на e-olymp.com

Вхідні дані

Натуральне число з проміжку від 10 до 99 включно.

Вихідні дані

Два однозначних числа, розділених пропуском.

Тестовий приклад

Вхідні дані	Вихідні дані
23	2 3

Підказка.

Для «поділу» знаків використовуйте операції цілочисельного ділення "/" і знаходження залишку від ділення "%".

1.7. Алгоритм Евкліда знаходження НСД двох натуральних чисел

Завдання 1.11. *(алгоритм Евкліда)*

Знайти найбільший спільний дільник двох натуральних чисел a і b.

1-й варіант рішення

```
#include <iostream>
using namespace std;
int main () {
	int a, b;
	cin >> a >> b;
	while (a! = b) {
		if (a> b)
			a = a - b;
		else
			b = b - a;
	}
	cout << a << endl;
}
```

2-й варіант рішення

```
#include <iostream>
using namespace std;
int main () {
	int a, b;
	cin >> a >> b;
	while (a! = 0 && b! = 0) {
```

```
            if (a> b)
                a% = b;
            else
                b% = a;
        }
        cout << a + b << endl;
    }
```

Завдання 1.12. *(самостійна робота)*

Скоротити дану дріб.

Вхідні дані

Два натуральних числа: чисельник і знаменник дробу

Вихідні дані

Чисельник і знаменник скороченого дробу, записаного за допомогою знаку "/".

Тестовий приклад

Вхідні дані	Вихідні дані
24 60	2/5

Підказки.

1. Знайти НСД і потім поділити чисельник і знаменник дробу на НСД.
2. Вивід відповіді:

cout << a << '/' << b << endl;

Завдання для самостійної роботи

Завдання 1

Вирішити завдання з сайту e-olymp.com

43, 111, 127, 128, 130, 138, 248, 407, 500, 519, 752, 1154, 1183, 2385.

Звіт по самостійній роботі включає:

- Текстовий документ з титульної сторінкою;

- Перед рішенням завдання - коротка теоретична інформація з даної теми (якщо це необхідно);
- Умова задачі, код рішення з коментарями;
- Скріншот виконання програми.

Завдання 2

За варіантами

Варіант 1.

1. Обчислити суму парних натуральних чисел від n до 105.
2. Серед довільного ряду цілих чисел, які вводяться з клавіатури, визначити кількість натуральних, кратних 9.
3. q (1000) шт. цегли можна перевозити візками місткістю 100, 300, 400 і 500 шт. цегли. Отримати всі можливі варіанти перевезень.

Варіант 2.

1. Знайти тризначні числа від n до 800, що діляться одночасно на 16 і 24.
2. Дано дійсне x і натуральне n. Обчислити: (x-1)(x-2)(x-3)…(x-n).
3. Футбольний м’яч коштує s(65) грн. Отримати всі можливі варіанти оплати, якщо у покупця є 5-, 10- і 20-гривневі купюри.

Варіант 3.

1. Обчислити: y = sin1(sin1+sin2)(sin1+sin2+sin3)...(sin1+sin2+sin3 + ... + sin n)
2. Дано натуральне n. Обчислити: n + (n -1) + (n - 2) + ... +1.
3. Садівникові потрібно z(18) кг мінеральних добрив. Отримати всі можливі варіанти купівлі добрива, якщо в магазині продаються розфасовки по 5, 4 і 2 кг.

Варіант 4.

1. Обчислити суму квадратів непарних натуральних чисел від i до n, де i задане натуральне число, i<n-1.
2. Ввести з клавіатури n довільних цілих чисел і обчислити середнє арифметичне від’ємних чисел.

3. Повітроплавцеві потрібно заповнити воднем повітряну кулю місткістю x(17) куб. м балончиками по 1, 2 і 5 куб. м водню. Отримати всі можливі варіанти наповнення.

Варіант 5.

1. Знайти тризначні числа від 100 до n, що дорівнюють сумі кубів своїх цифр.
2. Дано дійсне x і натуральне n. Обчислити: y = sinx + sin2x + sin3x + … + sin nx
3. Шляховим майстрам потрібно прокласти r(190) м залізниці рейками по 8 і 10 м. Отримати всі можливі варіанти прокладання.

Варіант 6.

1. Обчислити суму натуральних чисел, кратних 3, що належать інтервалу від 1 до n.
2. Дано дійсне a і натуральне n. Обчислити: y=a(a+1)(a+2)…(a+n-1)
3. d(36) кг яблук потрібно розфасувати у пакети по 2, 4, 5 і 10 кг. Отримати всі можливі варіанти розфасування.

Варіант 7.

1. Обчислити суму натуральних чисел, кратних 5, що належать інтервалу від i до n, де i - деяке натуральне число, менше від n.
2. Дано дійсне x і натуральне n. Обчислити: y=sin(x)+sin(2x)+sin(3x)+….+sin(nx).
3. p(120) осіб потрібно посадити за 4- та 6-містні столики. Отримати всі можливі варіанти поєднання столиків.

Варіант 8.

1. Ввести з клавіатури n довільних цілих чисел і обчислити кількість додатних непарних з них.
2. Знайти суму квадратів чисел від 1 до n.
3. На перевезення a(800) кг овочів з бази підготовлено ящики. В них можна завантажити по 8, 10 і 15 кілограмів. Отримати всі можливі варіанти завантаження.

Варіант 9.

1. Ввести з клавіатури n довільних цілих чисел і обчислити суму додатних парних з них.
2. Для ремонту дороги потрібно завезти p(24) т щебню. В автопарку є самоскиди вантажопідйомністю 3, 4 і 6 т. Отримати всі можливі варіанти перевезення щебню.
3. t(200) л бензину потрібно розлити у баки місткістю 60, 45 і 25 літрів. Отримати всі можливі варіанти розливу.

Варіант 10.

1. Обчислити суму довільних чисел, що вводяться з клавіатури і кількість яких наперед невідома. Розрахунок припиняється, якщо введене число дорівнює одиниці або нулю.
2. Задано послідовність натуральних чисел від 1 до n. Знайти суму чисел, некратних трьом.
3. Вантаж s(200) т можна перевозити вантажівками по 3, 4 і 5 т. Отримати всі можливі плани перевезень.

Варіант 11.

1. Обчислити добуток натуральних парних чисел від 1 до введеного тризначного числа, кратних 3, але некратних 9.
2. Дано дійсне x і натуральне n. Обчислити: (x-1)(x-2)(x-3)…(x-n).
3. x(50) тюльпанів потрібно розфасувати у подарункові набори по 3, 5 і 7 квіток. Вивести на екран всі можливі варіанти букетів.

Варіант 12.

1. Обчислити добуток натуральних чисел, кратних 5, від i до n.
2. Дано натуральне n. Обчислити: y = cos1*cos2*cos3*cos4*...*cosn.
3. У магазині для пересилання поштою підготовлено n(60) книг. Посилки комплектують по 10, 15 і 20 книг. Отримати всі можливі варіанти комплектів.

Варіант 13.

1. Знайти добуток усіх натуральних парних чисел від 20 до n.

2. Дано дійсне a і натуральне n. Обчислити: y=a(a+1)(a+2)…(a+n-1)
3. y(30) л соку потрібно розлити в 2- і 3-літрові банки. Вивести на екран всі можливі варіанти розливу.

Варіант 14.

1. Знайти двозначні числа від n до 99, для яких число дорівнює сумі подвоєного квадрата першої цифри і квадрата другої цифри.
2. Знайти суму квадратів двозначних непарних чисел від 10 до n, які діляться на 3, та вказати їх кількість.
3. t(350) саджанців помідорів для продажу потрібно розфасувати в пакети по 30, 50 і 100 шт. Отримати всі можливі варіанти розфасування.

Варіант 15.

1. Знайти суму квадратів двозначних чисел від n до m, що діляться одночасно на 4 і 6.
2. Дано натуральне n. Обчислити: 1 + 2 + 3 + … + (n -1) + n.
3. m(400) т зерна можна перевозити вантажівками по 2, 4 і 6 т. Отримати всі можливі варіанти перевезень.

Варіант 16.

1. Знайти добуток двозначних чисел від 20 до n, що діляться одночасно на 5 і 3 та вивести їх на екран.
2. Дано дійсне a і натуральне n. Обчислити: y = a*(a-n)*(a-2n)*(a-3n)*…*(a-n*n).
3. Потрібно викласти k(100) м бордюру брусками по 1, 2 і 3 м. Отримати всі можливі варіанти прокладання.

Питання для самоконтролю

1. Яку функцію повинні містити всі програми на C ++?

1) program()
2) system()
3) start()

4) main()

2. Який з нижче перерахованих операторів, не є циклом в С ++

1) for

2) do while

3) while

4) repeat until

3. Відзначте цикл з пост умовою

1) do while

2) for

3) while

4) repeat until

4. В якому випадку буде надруковано 20 зірочок - *?

1) int i, n = 20;
for(i = 0; i <= n; i++) printf("*");

2) int i, n = 20;
for(i = n; i >= 0; i--) printf("*");

3) int i, n = 20;
for(i = 19; i < n; i--) printf("*");

4) int i, n = 20;
for(i = 0; i < n; n++) printf("*");

5. Яке значення, за замовчуванням, повертає програма операційній системі в разі успішного завершення?

1) 0

2) – 1

3) 1

4) 42

6. Якими знаками закінчується більшість рядків коду в С ++?

1); (крапка з комою)

2). (крапка)

3): (двокрапка)

4), (кома)

7. Вкажіть правильне визначення функції main відповідно до специфікації стандарту ANSI

1) void main(void)

2) int main(void)

3) int main()

4) void main()

8. Відзначте цикл з передумовою

1) while

2) do while

3) for

4) repeat until

9. Мову програмування С ++ розробив

1) Б'ярн Страуструп

2) Ніклаус Вірт

3) Дональд Кнут

4) Кен Томпсон

10. Яка з наступних записів є правильним коментарем в С ++?

1) / * коментар * /

2) ** Коментар **

3) {коментар}

4) * / Коментар * /

11. Які службові символи використовуються для позначення початку і кінця блоку коду на мові С ++?

1) < >

2) ( )

3) begin end

4) { }

12. Виберіть правильний варіант оголошення константної змінної в С ++, де type - тип даних, variable - ім'я змінної value - константне значення

1) const variable = value;

2) const type variable := value;

3) const type variable = value;

4) const variable == value;

13. Назва С ++ запропонував

1) Дональд Кнут

2) Б'ярн Страуструп

3) Кен Томпсон

4) Рік Массітті

14. Чому буде дорівнює змінна a, після виконання цього коду

int a; for (a = 0; a < 10; a++) {}; ?

1) 10

2) 1

3) 0

4) 9

15. Щоб підключити заголовний файл в програму на С ++, наприклад iostream, необхідно написати:

1) #include <> з iostream всередині дужок

2) include # iostream, h;

3) include (iostreamh)

4) #include <>; з iostream.h всередині дужок

16. Прості типи даних в С ++.

1) цілі - int, дійсні - float або double, символьні - string

2) цілі - bool, дійсні - float або double, символьні - string

3) цілі - int, дійсні - float або real, символьні - char

4) цілі - int, дійсні - float або double, символьні - char

17. Який з наступних операторів - оператор порівняння двох змінних?

1) =

2) equal

3) ==

4) :=

18. Який з перерахованих типів даних не є типом даних в С ++?

1) float

2) int

3) real

4) double

Література до розділу

1. Code::Blocks. [Електронний ресурс]. – Режим доступу: https://www.codeblocks.org/downloads/binaries/

2. MinGW-w64 - for 32 and 64 bit Windows/. [Електронний ресурс]. – Режим доступу: https://sourceforge.net/projects/mingw-w64/

3. Visual Studio. Лучшие в своем классе средства для разработчиков. [Електронний ресурс]. – Режим доступу: https://visualstudio.microsoft.com/ru/

4. Алгоритмы и структуры данных. [Електронний ресурс]. – Режим доступу: http://cppstudio.com/cat/293/

5. Алгоритмы. [Електронний ресурс]. - Режим доступу: e-maxx.ru/algo.

6. Алгоритмы в стандартной библиотеке C++. [Електронний ресурс]. - Режим доступу: https://ravesli.com/algoritmy-v-standartnoj-biblioteke-s/

7. Документация. [Електронний ресурс]. – Режим доступу: http://www.cplusplus.com/reference/string/basic_string/

8. Знакомство с C++ Builder. [Електронний ресурс]. - Режим доступу: http://www.firststeps.ru/cbuilder/r.php?1

9. ИНФОРМАТИКС. Изучение языка программирования. [Електронний ресурс]. - Режим доступу: informatics.mccme.ru;

10. Компиляторы и среды разработки языка C++. [Електронний ресурс]. – Режим доступу: https://server.179.ru/tasks/cpp/total/101.html

11. Макаров. Программирование и основы алгоритмизации. [Електронний ресурс]. – Режим доступу: http://window.edu.ru/resource/126/25126/files/nwpi223.pdf

12. Основы программирования для начинающих. [Електронний ресурс]. – Режим доступу: http://cppstudio.com/

13. Руководство по языку программирования C++. [Електронний ресурс]. – Режим доступу: https://metanit.com/cpp/tutorial/

14. Сайт для программистов e-olymp. [Електронний ресурс]. - Режим доступу: e-olymp.com.

15. Сайт для программистов Timus Online Judge. [Електронний ресурс]. - Режим доступу: acm.timus.ru.

16. Уроки программирования на языке C++. [Електронний ресурс]. - Режим доступу: ravesli.com/uroki-cpp.

17. Учебник по C++. [Електронний ресурс]. – Режим доступу: http://cppstudio.com/cat/274/

Розділ 2. Складові типи даних. Масиви

2.1. Одновимірні масиви

Теоретичні відомості

До складових типів даних відносяться масиви, рядки, записи, множини, списки та інші структури. Масив є найпростішим складовим типом. Він містить набір однотипних елементів, розташованих підряд в безперервному відрізку пам'яті.

У цьому розділі ми розглянемо одновимірні масиви. Кожен елемент масиву має номер, причому нумерація елементів починається з нуля.

При роботі з масивами спочатку виконується оголошення масиву на деяку кількість елементів. При цьому резервується пам'ять для всіх елементів масиву відповідно до їх типу.

1 етап. Оголошення одновимірного масиву

типЕлементів і'мяМасиву [розмірність];

Наприклад, int arr[8];

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]	arr[7]

Рис. 2.1. Зберігання в масиві 8 елементів типу int

Звернення до елементів масиву:

arr[0]; - перший елемент масиву

arr[7]; - останній елемент масиву

2 етап. Ініціалізація або присвоювання початкових значень елементам масиву.

Можна використовувати різні способи ініціалізації масиву:

1) Введення даних з клавіатури

```
for (int i = 0; i <n; i++) {
        cin >> arr[i];
}
```

2) Ініціалізація масиву безпосереднім завданням елементів із зазначенням розміру масиву або без вказівки розміру

```
int arr[8] = {5, 1, –2, 0, 3, 1, 1, 6};
int arr[] = {5, 1, –2, 0, 3, 1, 1, 6};
```

3) Заповнення елементів масиву за деякою формулою

```
for (int i = 0; i <8; i++) {
        arr[i] = 2* i;
}
```

4) Заповнення випадковими значеннями

```
for (int i = 0; i <n; i++) {
        arr[i] = rand();
}
```

Функція rand() генерує випадкові числа в діапазоні від 0 до RAND_MAX = 32767. (RAND_MAX, максимальне значення, яке залежить від компілятора). Щоб отримати числа в діапазоні 0 .. 99, пишемо arr[i] = rand() % 100; щоб отримати числа в діапазоні –50 .. 50, задаємо команду присвоювання arr [i] = rand() % 100 – 50;

Завдання 2.1. *(алгоритм)*

Знайти суму елементів лінійного масиву цілих чисел.

Вхідні дані

Кількість елементів масиву і лінійний масив цілих чисел.

Вихідні дані

Сума елементів даного масиву

Тестовий приклад

Вхідні дані	Вихідні дані
8 –2 0 –4 15 1 –5 7 23	35

1-й варіант рішення

```
#include <iostream>
using namespace std;
int main() {
	int n;
	cin >> n;
	int arr[n];	// оголошення масиву на n елементів типу int
	for (int i = 0; i <n; i++) {
		cin >> arr[i]; //заповнення елементів масиву з клавіатури
	}
	int sum = 0;
	for (int i = 0; i <n; i++) {
		sum += arr[i];	//обчислення суми елементів масиву
	}
	cout << sum << endl;
}
```

2-й варіант рішення (можна об'єднати етапи зчитування даних і обчислення суми)

```
#include <iostream>
using namespace std;
int main() {
	int n, sum = 0;
	cin >> n;
	int arr[n];
	for (int i = 0; i <n; i++) {
		cin >> arr[i];
		sum += arr[i];
	}
	cout << sum << endl;
}
```

Завдання 2.2. *(алгоритм)*

Знайти добуток елементів лінійного масиву цілих чисел.

//Самостійно визначте *Вхідні і Вихідні дані* та підготуйте *Тестові приклади*.

```
#include <iostream>
using namespace std;
int main() {
	int n, p = 1;
	cin >> n;
	int arr[n];
	for (int i = 0; i <n; i++) {
		cin >> arr[i];
		p *= arr[i];
	}
	cout << p << endl;
}
```

Завдання 2.3. *(алгоритм)*

Пошук максимального елемента в лінійному масиві.

//Самостійно визначте *Вхідні і Вихідні дані* та підготуйте *Тестові приклади*.

```
#include <iostream>
using namespace std;
int main() {
	int n, max_el;		// max_el - максимальний елемент
	cin >> n;
	int arr[n];
	for (int i = 0; i <n; i++) {
		cin >> arr[i];
	}
	max_el = arr[0];    //приймаємо за максимальний елемент arr [0]
	for (int i = 1; i <n; i++) {
		if (arr[i]> max_el)		//пошук максимального елемента
			max_el = arr[i];
	}
	cout << max_el << endl;
}
```

Завдання 2.4. *(алгоритм)*

Пошук максимального і мінімального елементів.

//Самостійно визначте *Вхідні і Вихідні дані* та підготуйте *Тестові приклади*.

```
#include <iostream>
using namespace std;
int main() {
	int n, max_el, min_el;
	cin >> n;
	int arr[n];
	for (int i = 0; i <n; i++) {
		cin >> arr[i];
```

```
        }
        max_el = arr[0] ; min_el = arr[0];
        for (int i = 1; i <n; i++) {
                if (arr[i]> max_el) // пошук максимального елемента
                        max_el = arr[i];
                if (arr[i] <min_el) // пошук мінімального елемента
                        min_el = arr [i];
        }
        cout << max_el << ' ' << min_el << endl;
}
```

Завдання 2.5. *(приклад рішення)*

Знаходження номера елемента із заданими властивостями. Знайти номер максимального елемента.

//Самостійно визначте *Вхідні і Вихідні дані* та підготуйте *Тестові приклади*.

1-й варіант рішення

```
#include <iostream>
using namespace std;
int main() {
        int n, max_el;
        cin >> n;
        int arr[n];
        for (int i = 0; i <n; i++) {
                cin >> arr[i];
        }
        max_el = arr[ 0];
        int k;
        for (int i = 1; i <n; i++) {
           if (arr[i]> max_el) {
              max_el = arr[i];
```

```
                k = i;
            }
        }
        cout << k << endl;
    }
```

2-й варіант рішення

```
#include <iostream>
using namespace std;
int main() {
    int n, k;
    cin >> n;
    nt arr[n];
    for (int i = 0; i <n; i++) {
        cin >> arr[i];
    }
    k = 0;
    for (int i = 1; i <n; i++) {
        if (arr[i] > arr[k]) {
            k = i;
        }
    }
    cout << k << endl;
}
```

2.2. Використання булевих змінних для дострокового виходу з циклу

Теоретичні відомості

Змінна логічного типу може приймати одно з двох значень true або false. Bool означає булевий або логічний тип.

```
bool flag;          //оголошення булевої змінної
flag = false;       //ініціалізація
```

Використання булевих змінних

3. Вирази, що приймають значення булевого типу, використовуються в якості умови в розгалуженні або циклі.

4. Логічні значення часто використовуються в якості значень, що повертаються в функціях. Назви таких функцій дуже часто починаються зі слів is (наприклад, isEqual) або has (наприклад, hasCommonDivisor).

5. Змінні булевого типу можна використовувати також для відстеження деякої події. Наприклад, «в масиві знайдено перший позитивний елемент».

Назва операції	Знак операції	Назва
диз'юнкція (логічне складання, «або»)	||	or
кон'юнкція (логічне множення, «і»)	&&	and
заперечення («ні»)	!	not

Таблиця 2.1. Операції, що визначені на множині bool.

Покажемо, як використання булевої змінної призводить до дострокового виходу з циклу.

Завдання 2.6. *(приклад рішення)*

У лінійному масиві знайти перший позитивний елемент і його номер.

Вхідні дані

Кількість елементів масиву і лінійний масив цілих чисел.

Вихідні дані

Номер першого позитивного елементу і сам елемент.

Тестовий приклад

Вхідні дані	Вихідні дані
8 –2 0 –9 5 1 –5 7 3	3 5

```
#include <iostream>
using namespace std;
int main() {
	int n, k;
	bool flag = false;
	cin >> n;
	int arr[n];
	for (int i = 0; i <n && !flag ; i++) {
	//умова !flag — для дострокового виходу з циклу
	   cin >> arr[i];
	   if (arr[i]> 0) {
		flag = true;  //знайдений 1-й позитивний елемент
		k = i;          //запам'ятовуємо в змінну k номер елемента
	   }
	}
	cout << k << ' ' << arr [k] << endl;
}
```

2.3. Двовимірні масиви. Алгоритми з двовимірними масивами

Теоретичні відомості

Двовимірний масив в C ++ розглядається як масив, що складається з лінійних масивів. Двовимірні масиви утворені за допомогою рядків і

стовпців, тому їх елементи мають дві координати: номер рядка та номер стовпчика.

Оголошення масивів:

```
int mass[3][4];      //двовимірний масив з елементів типу int
```

Заповнення масиву (ініціалізація)

1) перерахуванням елементів

```
int mass[3][4] = {{5, 1, 2, 7}, {–3, 1, –1, 6}, {0, 0, 2, 4}}
```

5	1	2	7
–3	1	–1	6
0	0	2	4

Рис.2.2. Двовимірний масив з елементів типу int

Звернення до елементів двовимірного масиву

```
mass[0][0];          // mass[0][0] = 5;
mass[2][3];          // mass[2][3] = 4;
```

2) введення даних з клавіатури

```
for (int i = 0; i <n; i++) {
        for (int j = 0; j <m; j++) {
                cin >> arr[i][j];
        }
}
```

3) виведення елементів двовимірного масиву

```
for (int i = 0; i <n; i++) {
        for (int j = 0; j <m; j++) {
                cout << arr[i][j] << '  ';
        }
        cout << endl;                //перехід на новий рядок
}
```

Масиви можуть мати і більшу кількість вимірювань (3, 4 і так далі). Наприклад,

```
bool arr[3][10][5]; //тривимірний масив з елементів типу bool
```

Завдання 2.7. *(приклад рішення)*

Дан двовимірний масив цілих чисел. Обчислити суму всіх додатних елементів масиву.

Вхідні дані

Розміри масиву (кількість рядків і кількість стовпців). Потім — сам масив цілих чисел.

Вихідні дані

Сума всіх додатних елементів масиву.

Тестовий приклад

Вхідні дані	Вихідні дані
3 4 –1 4 6 0 –2 0 –9 5 1 –5 7 3	26

```
#include <iostream>
using namespace std;
int main() {
	int n, m;
	cin >> n >> m;
	int array[n][m];
	for (int i = 0; i <n; i++)
		for (int j = 0; j <m; j++)
			cin >> array[i][j];
	int sum = 0;
	for (int i = 0; i <n; i++) {
		for (int j = 0; j <m; j++) {
			if (array[i][j]> 0) sum += array[i][j];
		}
	}
	cout << "Сума додатніх елементів =" << sum << endl;
}
```

Завдання 2.8. *(приклад рішення)*

Найти номери стовпчиків, в яких є тільки один додатний елемент.

//Самостійно визначте *Вхідні і Вихідні дані* та підготуйте *Тестові приклади*.

```
#include <iostream>
using namespace std;
int main() {
	int n, m;
	cin >> n >> m;
	int array[n][m];
	for (int i = 0; i <n; i++)
		for (int j = 0; j <m; j+ +)
			cin >> array[i][j];
	int k;
	for (int j = 0; j <m; j++) {
		k = 0;
		for (int i = 0; i <n; i++) {
			if (array[i][j] > 0) k ++;
		}
		if (k == 1) cout << j << ' ';
	}
	cout << endl;
}
```

2.4. Вказівники та посилання

Теоретичні відомості

Вказівник (pointer) являє собою змінну, значення якої є адресом комірки пам'яті.

Оголошення вказівника:

<тип даних> * <ім'я вказівника>;

```
int * ptr;              //вказівник на int
int value = 5;
ptr = &value;           //ініціалізація вказівника
```

Вказівнику присвоюється адреса &value, за якою лежить значення змінної value (Рис.)

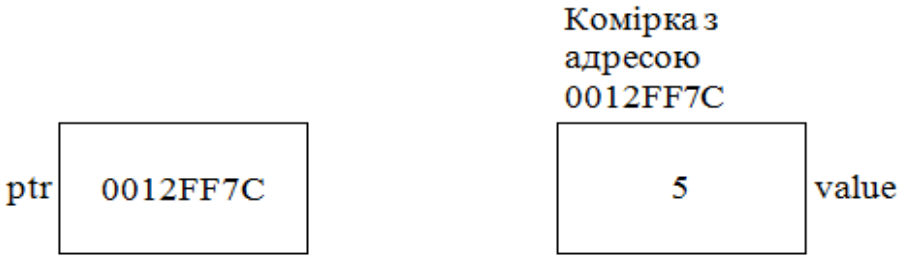


Рис. 2.3. ptr є вказівник на комірку з адресою 0012FF7C

Розіменування вказівника і отримання адреси

Використання вказівників тісно пов'язане із застосуванням двох операцій: розіменування вказівника і отримання адреси змінної.

Оператор розіменування вказівника * дозволяє отримати значення за вказаною адресою. int Num = * ptr; // Num = 5;

Тобто, можна змінювати або читати значення змінної через вказівник.

```
cout << ptr;                    // 0012FF7C
cout << * ptr;                  // 5
cout << value;                  // 5
```

Вказівники дозволяють отримати безпосередній доступ до пам'яті.

Оператор взяття адреси &

```
int x = 10, y = 20;
cout << "Value x =" << x << endl;          // Value x = 10
cout << "Adress x =" << &x << endl;        // Adress x = 0x0012FF7C
cout << "Value y =" << y << endl;          // Value y = 20
cout << "Adress y =" << &y << endl;        // Adress x = 0x0012FF78
```

Отриманням адреси змінної називається конструкція, що дозволяє сформувати вказівник, який зберігає адресу цієї змінної.

У певному сенсі описані операції є зворотними по відношенню один до одного.

Додаткові посилання

[9], [11].

2.5. Динамічні масиви

Теоретичні відомості

Оголошення одновимірного динамічного масиву

Інструкція

```
int * mass = new int[n];
```

створює одновимірний масив в динамічній пам'яті.

```
delete [] mass;                    //оператор delete очищає пам'ять
```

(Про види пам'яті буде далі, в розділі 4).

Оголошення двовимірного динамічного масиву

```
int **arr = new int *[n];
for (int i = 0; i <n; i++) {
        arr[i] = new int[m];
}
delete [][] arr;                   // очищення пам'яті
```

Завдання 2.9. *(приклад рішення)*

Виконати множення квадратних матриць розміром n x n

//Самостійно визначте *Вхідні і Вихідні дані* та підготуйте *Тестові приклади*.

```
#include <iostream>
using namespace std;
int main() {
        int n;
        cin >> n;
        int **m1 = new int *[n];
        for (int i = 0; i <n; i++)
```

```
        m1[i] = new int[n];
    int **m2 = new int *[n];
    for (int i = 0; i <n; i++)
        m2[i] = new int[n];
    for (int i = 0; i <n; i++) {
        for (int j = 0; j <n; j++) {
            m1[i][j] = 1; // 1-й масив заповнюється одиницями
            m2[i][j] = 2; // 2-й масив заповнюється двійками
        }
    }
    int ** res = new int *[n];
    for (int i = 0; i <n; i++)
        res[i] = new int[n];
    for (int i = 0; i <n; i++) {
        for (int j = 0; j <n; j++) {
            res[i][j] = 0;
            for (int p = 0; p <n; p++)
                res[i][j] += m1[i][p] * m2[p][j];
        }
    }
    delete [][]m1;
    delete [][]m2;
    for (int i = 0; i <n; i++) {
        for (int j = 0; j <n; j++) {
            cout << res[i][j] << '  ';
        }
        cout << endl;
    }
    delete [][]res;
    return 0;
}
```

2.6. Алгоритми сортування

Теоретичні відомості

Сортування це упорядкування елементів в масиві або списку за зростанням або спаданням.

Види сортування

- Сортування бульбашками
- Сортування вставками
- Сортування підрахунком
- Сортування злиттям
- Сортування вибором
- Сортування Шелла (sort)
- Швидке сортування (qsort)
- Пірамідальне сортування
- Сортування за допомогою дерева

Прикладів алгоритмів сортування масивів можна навести кілька десятків. Додаткову інформацію можна отримати тут [1], [2], [5].

Ми розглянемо спочатку найпростіше сортування – сортування бульбашками.

Словесне формулювання алгоритму: Наступний елемент масиву по черзі порівнюємо з усіма іншими. Якщо вони стоять не за порядком (не за зростанням), то міняємо їх місцями.

Завдання 2.10. *(приклад рішення)*

Виконати сортування лінійного масиву за зростанням.

Вхідні дані

Кількість елементів масиву і масив цілих чисел.

Вихідні дані

Відсортований масив.

Тестовий приклад

Вхідні дані	Вихідні дані
8 –2 0 –9 5 1 –5 7 3	–9 –5 –2 0 1 3 5 7

```
int main() {
	int n;
	cin >> n;
	int * arr = new int[n];
	for (int i = 0; i <n; i++) {
		cin >> arr[i];
	}
	for (int i = 0; i < n – 1; i++) {
		for (int j = i + 1; j < n; j++) {
			if (arr[j] < arr[i])
				swap(arr[j], arr[i]);
		}
	}
	for (int i = 0; i < n – 1; i++) {
		cout << arr[i] << ' '; //виведення елементів, крім останнього
	}
	cout << arr[n - 1] << endl;	//виведення останнього елемента
}
```

Сортування вибором

1. Знаходимо номер мінімального значення в поточному списку

2. Проводимо обмін цього значення із значенням першого невідсортованого елемента (обмін не потрібний, якщо мінімальний елемент вже знаходиться на даній позиції)

3. Тепер сортуємо хвіст списку, виключивши з розгляду вже відсортовані елементи

```
#include <iostream>
using namespace std;
int main() {
   int len;
   cin >> len;
   int *a;
   a = new int[len];
   for (int i = 0; i < len; i++) {
      cin>>a[i];
   }
/ * Зовнішній цикл. i - позиція першого невідсортованого елемента на даній ітерації * /
  for (int i = 0; i <len - 1; i ++) {
    int min = i;              / * min - позиція мінімального елемента * /
    / * Внутрішній цикл. Якщо знайдений елемент строго менший поточного мінімального, записуємо його індекс як мінімального * /
    for (int j = i + 1; j < len; j++) {
      if (a[j] <a[min])
        min = j;
    }
    if (min! = i) / * мінімальний елемент не є першим несортованим, обмін потрібен * /
      swap(a[i], a[min]);
  }
  for (int l = 0; l < len - 1; l++) {
    cout << a[l] << ' ';
  }
  cout << a[len - 1] << endl;
  return 0;
}
```

Алгоритм. Швидке сортування (Quicksort)

qsort – функція в стандартній бібліотеці мови С. Вона реалізує широко відомий алгоритм сортування , розроблений Чарльзом Хоаром в 1960 році. Це один із найшвидших відомих універсальних алгоритмів сортування масивів. Його складність в середньому O(n*log n) при впорядкуванні n елементів.

Алгоритм

1. Вибираємо в масиві деякий елемент , який будемо називати опорним елементом . Наприклад, середній .

2. Операція поділу масиву: все елементи , які менше або рівні опорному елементу, повинні розміститися зліва від нього , а все елементи, які більше опорного – праворуч від нього.

- Два індексу first і last набувають значень відповідно мінімального і максимального індексів.
- Визначається опорний елемент p: = arr [(last-first) div 2 + first];
- Індекс first збільшується до тих пір, поки і-й елемент НЕ перевищить опорний.
- Індекс last послідовно зменшується до тих пір, поки j-й елемент НЕ виявиться меншим або рівним опорному елементу.
- Якщо first = last, то знайдена середина масиву. Тоді операція поділу закінчена, обидва індексу first і last вказують на опорний елемент .
- Якщо first < last, то знайдену пару елементів потрібно обміняти місцями і продовжити операцію поділу з тих значень first і last , які були досягнуті.

3. Рекурсивне упорядковуємо підмасиви, що розміщені ліворуч і праворуч від опорного елементу.

4. На кожному наступному кроці рекурсії довжина відрізка масиву зменшується, щонайменше, на одиницю.

Завдання 2.11. *(алгоритм)*

Виконати сортування за зростанням даного масиву методом швидкого сортування

```
int *arr;
void qs(int arr[], int first, int last) {
   int p=arr[(first+last)/2];
   int i=first;
   int j=last;
   while (i<=j) {
      while (arr[i]<p) i++;
      while (arr[j]>p) j--;
      if (i<=j) {
         swap(arr[i],arr[j]);
         i++;
         j--;
      }
      if (j>first) qs(arr,first,j);
      if (i<last) qs(arr,i,last);
   }
}
int main() {
   int n;
   cin >> n;
   arr = new int [n];
   for (int i = 0; i < n; i++) {
      cin>>arr[i];
   }
   qs(arr,0,n-1);
   for (int i = 0; i < n; i++) {
      cout << arr[i] << ‘ ’;
```

```
    }
    cout << endl;
}
```

Додаткові посилання

[1], [6].

Завдання для самостійної роботи

За варіантами

Варіант 1.

4. Дано двовимірний масив цілих чисел. Всі додатні елементи заданого масиву розташувати в порядку зростання методом обміну.
5. Дано лінійний масив цілих чисел. Відсортувати його методом вибору і вставки та визначити кількість порівнянь і який метод ефективніше.

Варіант 2.

4. Дано двовимірний масив цілих чисел. Всі непарні елементи заданого масиву розташувати в порядку зростання методом вставки.
5. Дано лінійний масив цілих чисел. Відсортувати його методом обміну і виділення та визначити кількість порівнянь і який метод ефективніше.

Варіант 3.

4. Дано двовимірний масив цілих чисел. Всі елементи масиву, що при діленні на 4 дають в остачі 2, розташувати в порядку зростання методом вибору.
5. Дано лінійний масив цілих чисел. Відсортувати його методом бульбашки і вставки та визначити кількість порівнянь і який метод ефективніше.

Варіант 4.

4. Дано двовимірний масив цілих чисел. Всі парні елементи заданого масиву розташувати в порядку спадання методом включення.
5. Дано лінійний масив цілих чисел. Відсортувати його методом вибору

і бульбашки та визначити кількість порівнянь і який метод ефективніше.

Варіант 5.

4. Дано двовимірний масив цілих чисел. Всі від’ємні елементи заданого масиву розташувати в порядку спадання методом «бульбашок».
5. Дано лінійний масив цілих чисел. Відсортувати його методом вибору і включення та визначити кількість порівнянь і який метод ефективніше.

Варіант 6.

4. Дано двовимірний масив цілих чисел. Всі елементи заданого масиву, що не кратні 3, розташувати в порядку зростання методом виділення.
5. Дано лінійний масив цілих чисел. Відсортувати його методом обміну і включення та визначити кількість порівнянь і який метод ефективніше.

Варіант 7.

4. Дано двовимірний масив цілих чисел. Всі елементи заданого масиву, що більше 3, розташувати в порядку спадання методом вибору.
5. Дано лінійний масив цілих чисел. Відсортувати його методом включення і обміну та визначити кількість порівнянь і який метод ефективніше.

Варіант 8.

4. Дано двовимірний масив цілих чисел. Всі непарні елементи заданого масиву розташувати в порядку спадання методом обміну.
5. Дано лінійний масив цілих чисел. Відсортувати його методом виділення і вставки та визначити кількість порівнянь і який метод ефективніше.

Варіант 9.

4. Дано двовимірний масив цілих чисел. Всі елементи заданого масиву, що більші 10, розташувати в порядку зростання методом включення.
5. Дано лінійний масив цілих чисел. Відсортувати його методом вибору

і бульбашки та визначити кількість порівнянь і який метод ефективніше.

Варіант 10.

4. Дано двовимірний масив цілих чисел. Всі додатні елементи заданого масиву розташувати в порядку зростання методом вибору.
5. Дано лінійний масив цілих чисел. Відсортувати його методом обміну і включення та визначити кількість порівнянь і який метод ефективніше.

Варіант 11.

4. Дано двовимірний масив цілих чисел. Всі від'ємні елементи заданого масиву розташувати в порядку спадання методом виділення.
5. Дано лінійний масив цілих чисел. Відсортувати його методом бульбашки і вставки та визначити кількість порівнянь і який метод ефективніше.

Варіант 12.

4. Дано двовимірний масив цілих чисел. Всі від'ємні елементи заданого масиву розташувати в порядку спадання методом вставки.
5. Дано лінійний масив цілих чисел. Відсортувати його методом вибору і обміну та визначити кількість порівнянь і який метод ефективніше.

Варіант 13.

4. Дано двовимірний масив цілих чисел. Всі парні додатні елементи заданого масиву розташувати в порядку зростання методом «бульбашок».
5. Дано лінійний масив цілих чисел. Відсортувати його методом вибору і вставки та визначити кількість порівнянь і який метод ефективніше.

Варіант 14.

4. Дано двовимірний масив цілих чисел. Всі елементи заданого масиву, що дають при діленні на 7 остачу 5, розташувати в порядку зростання методом включення.
5. Дано лінійний масив цілих чисел. Відсортувати його методом обміну і виділення та визначити кількість порівнянь і який метод ефективніше.

Варіант 15.

4. Дано двовимірний масив цілих чисел. Всі елементи заданого масиву, що менші 5, розташувати в порядку зростання методом обміну.
5. Дано лінійний масив цілих чисел. Відсортувати його методом вибору і включення та визначити кількість порівнянь і який метод ефективніше.

Варіант 16.

- Всі парні елементи заданого масиву розташувати в порядку спадання методом виділення.
- Дано лінійний масив цілих чисел. Відсортувати його методом бульбашки і вставки та визначити кількість порівнянь і який метод ефективніше.

Варіант 17.

- Дано двовимірний масив цілих чисел. Всі елементи заданого масиву, які менші 7, але більші -3 розташувати в порядку спадання методом вставки.
- Дано лінійний масив цілих чисел. Відсортувати його методом обміну і вибору та визначити кількість порівнянь і який метод ефективніше.

Варіант 18.

1) Дано двовимірний масив цілих чисел. Всі елементи заданого масиву, що кратні 4, розташувати в порядку спадання методом включення.
2) Дано лінійний масив цілих чисел. Відсортувати його методом бульбашки і виділення та визначити кількість порівнянь і який метод ефективніше.

Варіант 19.

1) Дано двовимірний масив цілих чисел. Всі додатні парні елементи заданого масиву, розташувати в порядку спадання методом вибору.
2) Дано лінійний масив цілих чисел. Відсортувати його методом обміну і вставки та визначити кількість порівнянь і який метод ефективніше.

Варіант 20.

1) Дано двовимірний масив цілих чисел. Всі елементи заданого масиву, які є точними квадратами, розташувати в порядку зростання методом «бульбашок».
2) Дано лінійний масив цілих чисел. Відсортувати його методом виділення і включення та визначити кількість порівнянь і який метод ефективніше.

Варіант 21.

1) Дано двовимірний масив цілих чисел. Всі елементи заданого масиву, що більші -5, розташувати в порядку зростання методом вибору.
2) Дано лінійний масив цілих чисел. Відсортувати його методом обміну і вставки та визначити кількість порівнянь і який метод ефективніше.

Варіант 22.

1) Дано двовимірний масив цілих чисел. Всі елементи заданого масиву, що не кратні 7, розташувати в порядку зростання методом вставки.
2) Дано лінійний масив цілих чисел. Відсортувати його методом бульбашки і виділення та визначити кількість порівнянь і який метод ефективніше.

Варіант 23.

1) Дано двовимірний масив цілих чисел. Всі непарні елементи заданого масиву, що менші 10, розташувати в порядку спадання методом виділення.
2) Дано лінійний масив цілих чисел. Відсортувати його методом обміну і включення та визначити кількість порівнянь і який метод ефективніше.

Варіант 24.

1) Дано двовимірний масив цілих чисел. Всі елементи заданого масиву, що кратні 3, розташувати в порядку спадання методом вставки.
2) Дано лінійний масив цілих чисел. Відсортувати його методом вибору і бульбашки та визначити кількість порівнянь і який метод ефективніше.

Варіант 25.

4. Дано двовимірний масив цілих чисел. Всі елементи заданого масиву, що не є точними квадратами, розташувати в порядку спадання методом включення.
5. Дано лінійний масив цілих чисел. Відсортувати його методом виділення і обміну та визначити кількість порівнянь і який метод ефективніше.

Питання для самоконтролю

1. Коректне виділення пам'яті
   1) int * a = new sizeof (int * 20);
   2) int * a = new int [20];
   3) int * a = new 20;
   4) int a = new int [20];
2. В якому з варіантів відповідей оголошений двовимірний масив?
   1) array anarray [20] [20];
   2) int anarray [20] [20];
   3) char array [20];
   4) int array [20, 2 0];
3. Вкажіть правильне оголошення масиву
   1) int anarray;
   2) array an array [10];
   3) int anarray [10];
   4) anarray {10};
4. Що таке посилання?
   1) немає правильної відповіді
   2) посилання є псевдонімом для об'єкта
   3) використовується для перейменування об'єктів
   4) те саме, що і вказівник

5. Як правильно звільнити пам'ять, після використання масива?

1) char * a; a = new char [20];
2) delete a;
3) delete a [];
4) delete [] a;

6. Масив - це ...

1) Масив - це впорядковані в пам'яті елементи одного і того ж типу, що мають загальну адресу. Доступ до окремих елементів масиву здійснюється за адресою і індексом
2) Масив - це впорядковані в пам'яті елементи одного і того ж типу, що мають ім'я. Доступ до окремих елементів масиву здійснюється за ім'ям масиву і індексом
3) Масив - це впорядковані в пам'яті елементи одного і того ж типу, що мають ім'я. Доступ до окремих елементів масиву здійснюється за ім'ям масиву і адресою

7. Вкажіть зарезервоване ключове слово для вивільнення виділеної пам'яті

1) delete
2) clear
3) remove
4) free

8. Задано масив символів

```
char arr [8];
cin >> arr;
```

В масив arr ми спробували записати наступний набір символів Hello World. Що насправді буде містити масив arr?

1) Hello
2) Hello Wo
3) Hello W
4) Hello World

9. В якому з наступних записів використовується операція розіменування?

1) address (a);

2) a;

3) & a;

4) * a;

10. Вкажіть рядок, який повертає адресу першого елемента в масиві arr?

1) arr [1];

2) arr;

3) & arr;

4) arr [0];

11. Який порядковий номер останнього елемента масиву, розмір масиву 19?

1) 19

2) 18

3) 20

4) порядковий номер визначається програмістом

12. Словосполучення "Hello world!" може бути збережено в символьному масиві розміром n елементів. Вкажіть чому дорівнює n?

1) 11

2) 13

3) 12

4) 10

13. Який з наступних записів повертає значення змінної a, що зберігається в пам’яті за адресою на яку вказує вказівник?

1) val (a);

2) * a;

3) & a;

4) a;

14. Яке значення буде надруковано, в результаті виконання наступного коду?

```
#include <iostream>
```

```
int main() {
        int sum = 0;
        int array[3][] = {{0, 1, 2}, {3, 4, 5}, {6, 7, 8}};
        for (int i = 0; i <3; ++ i) {
                for (int j = 2; j <3; j ++) {
                        sum + = array[i][j];
                }
        }
        std :: cout << sum << std :: endl;
        return 0;
}
```

1) 15
2) 12
3) 9
4) синтаксична помилка

15. Після виконання ряду операцій з вказівником, що буде виведено на екран?

```
int main (int argc, char ** argv) {
        // припустимо, int займає 4 байта
        std :: cout << sizeof(int) << std :: endl;
        int * x = new int;
        // припустимо адреса дорівнює 0x60450000
        std :: cout << x << std :: endl;
        std :: cout << x + 3 << std :: endl;
        return 0;
}
```

1) 0x6045000C
2) некоректне визначення
3) 0x60450003
4) 0x60450000

16. Вкажіть правильне оголошення вказівника в C ++

1) int & x;

2) int * x;

3) int x;

4) ptr x;

17. В якій з наступних записів використовується операція взяття адреси?

1) address(a);

2) *a;

3) &a;

4) a;

18. В якій з наступних рядків виконується звернення до сьомого елементу масиву, розмір масиву дорівнює 10?

1) mas[7];

2) mas(7);

3) mas;

4) mas[6];

Література до розділу

18. Алгоритмы сортировки: реализация на C++. [Електронний ресурс]. – Режим доступу: https://function-x.ru/cpp_algoritmy_sortirovki.html

19. Вирт Н. Алгоритмы и структуры данных. - М.: Мир. 1989. - 360 с.

20. Дейтел, Дейтел. Как программировать на C++: Пятое издание. М.: Издательство «Бином-Пресс», 2008. - 1456с. (рос.)

21. Мейерс С. Эффективное использование STL. Библиотека программиста. - Спб.: Питер, 2002. - 224 с. (рос.)

22. Описание алгоритмов сортировки и сравнение их производительности. [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/335920/

23. Решето Эратосфена. [Електронний ресурс]. – Режим доступу:

https://purecodecpp.com/archives/2517

24. Сайт для программистов e-olymp. [Електронний ресурс]. - Режим доступу: e-olymp.com.
25. Сайт для программистов Timus Online Judge. [Електронний ресурс]. - Режим доступу: acm.timus.ru.
26. Указатели. [Електронний ресурс]. – Режим доступу: https://younglinux.info/c/function
27. Уроки программирования на языке C++. [Електронний ресурс]. - Режим доступу: ravesli.com/uroki-cpp.
28. Учебник по C++. [Електронний ресурс]. – Режим доступу: http://cppstudio.com/cat/274/

Розділ 3. Рядки, їх реалізація. Алгоритми на рядках.

3.1. Рядки, як масив символів

Теоретичні відомості

Існує два підходи в роботі з рядками.

Розглянемо перший підхід, який передбачає завдання рядків у вигляді масиву символів, що було реалізовано в мові С.

Функції для роботи з рядками:

```
gets(s);                 //читання всього тексту з клавіатури
puts(s);                 //виведення тексту на екран
strcpy(str , "World");   //копіювання 2-го аргументу в рядок str
strlen(s)                //обчислення довжини рядка s
strcat(s1, s2);          //конкатенація рядків. Результат – в s1
strchr(s, x);            //пошук першої появи символу x в рядку s
```

та інші.

Завдання 3.1. *(приклад рішення)*

Конкатенація (додавання) рядків

```
#include <stdio.h>                    // gets , puts
#include <string.h>
int main() {
        const int n = 10;
        char s1[n];
        const char s2[] = "aaaaa";
        gets(s1);
        strcat(s1, s2);
        puts(s1);
        return 0;
}
```

Завдання 3.2. *(приклад рішення)*

Копіювання рядків

```
#include <stdio.h>
#include <string.h>
int main() {
        char str[80];
        strcpy(str , "this is an example");
        puts(str);
        return 0;
}
```

Завдання 3.3. *(приклад рішення)*

Підрахувати кількість цифр цілого невід'ємного числа n.

```
#include <iostream>
using namespace std;
int main() {
char s[20];
cin >> s;
cout << strlen(s) << endl;
```

```
return 0;
}
```

3.2. Шаблонний строковий клас string

Теоретичні відомості

В С ++ для подання рядків можна використовувати як масиви символів, так і клас *string*. Він визначений як шаблон класу в просторі імен *std* в заголовку <string> . Це другий підхід при роботі з рядками.

Завдання 3.4. *(приклад рішення)*

У програмі оголошена змінна name типу string. З клавіатури вводиться ім'я і потім виводиться текст "Hello, name".

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string name;
    cin >> name;
    cout << "Hello," << name << endl;
}
```

Працювати з рядками через клас string досить зручно – можна робити конкатенацію (додавання) рядків за допомогою звичайного оператора +, можна брати символ в певному місці рядка за допомогою оператора [] або за допомогою методу at(), можна використовувати звичайні оператори =, ==, != для присвоювання значення і порівняння рядків. Також є методи для отримання довжини рядка, для з'ясування, чи не порожній рядок та ін. Корисним є метод getline, який дозволяє прочитати весь текст із вхідного потоку.

3.3. Функції для роботи зі змінними типу string

Теоретичні відомості

- s.append(str) - додає в кінець рядка рядок str.
- s.assign(str) – присвоює рядку s значення рядка str,
- аналогічно запису s = str;
- int i = s.begin() – індекс першого елемента рядка;
- int i = s.end() – індекс останнього елемента рядка;
- s.clear() – очищає рядок;
- s.compare(str) – порівнює рядок s з рядком str і повертає 0 в разі збігу (насправді порівнює коди символів і повертає їх різницю);
- s.copy(str, len, pos) – копіює з рядка s в str, кількість символів (len), починаючи з деякої позиції (pos);
- bool b = s.empty() – якщо рядок порожній, повертає true, інакше – false;
- s.erase() – видаляє n елементів із заданою позиції;
- s.find(str, pos) – шукає підрядок str в рядку s, починаючи з заданої позиції;
- s.find('w') – знаходить позицію символу 'w';
- s.insert(pos, str , begin, count) – вставляє в рядок s починаючи з заданої позиції pos частину рядка str, починаючи з позиції begin, count символів;
- int len = s.length() – записує в len довжину рядка;
- s.push_back(symbol) – додає в кінець рядка символ;

- s.replace(index, n, str) – бере n перших символів з str і замінює символи рядка s на них, починаючи з позиції index;
- str = s.substr(n, m) - повертає m символів починаючи з позиції n;
- s.size() - повертає число елементів в рядку;
- s.at(1); - отримуємо символ з рядка s, з позиції 1;
- swap(str1, str2); - обмін значеннями двох рядків;
- int num= int(c) - отримання номера символу **c;**
- char c = chr(n) - отримання символу з номером **n.**

Як встановити параметри символів, що залежать від регіону (мовна настройка)

```
setlocale (LC_ALL, "Ukrainian");   або
setlocale (0, "");
```

Приклади використання функцій:

1. Отримати символ з номером 1 з рядка s.

```
string s = "abcdef";
char ch = s.at(1);              //отримаємо b
```

2. Перевірити, чи не порожній рядок.

```
if (s.empty())
    cout << "String is empty" << endl;
    else
    cout << "String is not empty" << endl;
```

3. Задаємо і порівнюємо два слова.

```
string  s0, s1;
cin >> s0 >> s1;
if (s1 == s0)
        cout << " the  words match " << endl;
        else
        cout << "the words don't match " << endl;
```

4. Отримання довжини рядка.

```
cout << s2.length() << endl;
```

Зауваження.

getline(cin, s); читає весь рядок (текст),

cin >> s; читає частину рядка s, до першого символу-роздільника.

Завдання 3.5. *(приклад рішення)*

Конкатенація (склеювання) двох рядків і знаходження довжини рядка результату.

```
#include <iostream>
#include <string>
using namespace std;
int main() {
        string s1 = "Hello";
        string s2 = "world";
        string s3 = s1 + "," + s2 + "\ n";
        cout << s3 .length() << endl;              // довжина рядка – 12
}
```

Завдання 3.6. *(приклад рішення)*

Отримання рядка з вхідного потоку (з клавіатури).

```
#include <iostream>
using namespace std;
int main() {
	string s1, s2;
	cout << "Enter string s1:";
	getline(cin, s1);		// getline читає весь текст
	s2 = s1;
	cout << "s1 =" << s1 << endl;
	cout << "s2 =" << s2;
	return 0;
}
```

Завдання 3.7. *(алгоритм). Пошук підрядка в рядку.*

Чи є підрядок short префіксом рядка longer?

```
#include <iostream>
#include <string>
using namespace std;
int main() {
	bool isPrefix = false;
	string longer, short;
	getline(cin, longer);
```

```
        getline(cin, short);
        if ( longer.substr(0, short.length ()) == short)
            isPrefix = true;
        cout << isPrefix << endl;
}
```

Завдання 3.8. *(приклад рішення)*

Задача 494 на сайті e-olymp.com.

До гласних букв в латинському алфавіті відносяться літери A, E, I, O, U і Y. Інші літери вважаються приголосними. Напишіть програму, яка підраховує кількість голосних букв в тексті.

Вхідні дані

У вхідному потоці - один рядок тексту, що складається тільки з великих латинських букв і пробілів. Довжина рядка не перевищує 100 символів.

Вихідні дані

Вивести одне ціле число - кількість голосних у вхідному тексті.

Вхідні дані	Вихідні дані
FIND ALL VOWELS IN A TEXT	7

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    int k = 0;
```

```
        string vowel = "AEIOUY";
        string text;
        getline(cin, text);
        for (int j = 0; j <6; j ++) {
                for (int i = 0; i <text.length (); i ++)
                        if (text [i] == vowel[j])
                                k ++;
        }
        cout << k << endl;
}
```

Завдання 3.9. *(самостійна робота)*

Задача 909 на e-olymp.com.

Визначити кількість слів у заданому фрагменті тексту

Завдання 3.10. *(приклад рішення)*

Змінній ch типу char присвоюється значення символу 'A' . Потім на екран виводиться значення змінної ch та її ASCII-код .

Змінній ch типу char можна присвоїти і числове значення 97. Відбувається присвоювання символу з кодом 97. Знову виводимо на екран символ і його ASCII-код.

```
#include <iostream>
using namespace std;
int main() {
char ch = 'A';          // один символ береться в одинарні лапки
cout << ch << " " << (int) ch << endl;
ch = 97 ;               // char можна привласнити і числове значення
cout << ch << "  " << (int)ch << endl;
}
```

Поради

1. При роботі з рядками виберіть один з підходів і користуйтеся ним.

2. Зручніше користуватися string.

3.4. Алгоритми з рядками та послідовностями

Теоретичні відомості

Читання даних з вхідного потоку

```
char c ;
while (cin >> c)              // доки зчитування успішно
{
// Робимо необхідні дії
}
```

У цьому прикладі програма буде посимвольно зчитувати вхідний потік, поки не зустріне ознаку кінця файлу. Для того щоб повідомити програму про завершення вводу при введенні з клавіатури, необхідно натиснути клавіші Ctrl-d в системі Linux або Ctrl-z в системі Windows.

Інший варіант – передбачити появу певного символу. Наприклад, символу " !", як в наступній задачі.

Завдання 3.11. *(приклад рішення)*

Знайти найдовше слово в тексті.

```
#include <iostream>
#include <string>
using namespace std;
int main() {
        string s, word;
        cin >> s;
        int max_len = s.length ();
```

```
        word = s;
        while (s ! = "!") {                    //поки не прочитали символ "!"
            cin >> s;
            if (s.length() > max_len) {
                max_len = s.length();
                word = s;
            }
        }
        cout << word << endl;
        return 0;
}
```

Завдання 3.12. *(приклад рішення)*

Знайти максимальну довжину підпослідовності однакових елементів.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main() {
        int n, a, a0, max_length = 1, length = 1;
        cin >> n;
        cin >> a0;
        int i = 1;
        while (i < n) {
            cin >> a;
```

```
            while (a == a0) {
                length++;
                i++;
                a0 = a;
                if (i < n)
                    cin >> a;
                else
                    break;
            }
            if (length > max_length) {
                max_length = length;
            }
            length = 1;
            i++;
            a0 = a;
        }
        cout << max_length << endl;
        return 0;
}
```

Завдання 3.13. *(приклад рішення)*

Дан масив з n чисел: Потрібно знайти в цій послідовності строго зростаючу підпослідовність найбільшої довжини.

Рішення за O(n2): метод динамічного програмування

Перший крок. Алгоритм знаходить і виводить довжину наідліннішої зростаючої підпослідовності:

```
int d [MAXN];              // MAXN - найбільше можливе значення n

for (int i=0; i<n; ++i) {
        d[i] = 1;
        for (int j=0; j<i; ++j)
                if (a[j] < a[i])
                        d[i] = max (d[i], 1 + d[j]);
}


int ans = d[0];
for (int i=0; i<n; ++i)
        ans = max(ans, d[i]);
cout << ans << endl;
```

Другий крок. Відновлення відповіді.

Будемо зберігати допоміжний масив з індексами, при яких вийшло найбільше значення. Цей масив будемо називати "масив предків". Тоді, щоб вивести відповідь, треба просто йти від елемента з максимальним значенням за його предками до тих пір, поки ми не виведемо всю підпослідовність, тобто поки не дійдемо до елемента зі значенням 1.

Додамо також код виведення наідліннішої підпослідовності.

```
int d [MAXN], p [MAXN];

for (int i=0; i<n; ++i) {
        d[i] = 1;
        p[i] = -1;
        for (int j=0; j<i; ++j)
                if (a[j] < a[i])
                        if (1 + d[j] > d[i]) {
                                d[i] = 1 + d[j];
```

```
                        p[i] = j;
                    }
        }
        int ans = d[0],  pos = 0;
        for (int i=0; i<n; ++i)
            if (d[i] > ans) {
                ans = d[i];
                pos = i;
            }
        cout << ans << endl;
        vector<int> path;
        while (pos != –1) {
            path.push_back (pos);
            pos = p[pos];
        }
        for (int i(int)path.size() – 1; i >=0; --i)
            cout << path[i] << ' ';
```

Альтернативний спосіб відновлення відповіді.

Заново перераховуємо поточний елемент і шукаємо, на якому індексі досягається максимум.

Рішення за O (n*log n): динамічне програмування з двійковим пошуком.

Скориставшись стандартним алгоритмом двійкового пошуку, який повертає позицію першого елемента, строго більшого даного елементу, отримуємо таку реалізацію:

```
int d[MAXN];
d[0] = INF;
for (int i=1; i<=n; ++i)
    d[i] = INF;
```

```
for (int i=0; i<n; i++) {
        int j = int (upper_bound (d.begin(), d.end(), a[i]) - d.begin());
        if (d[j-1] < a[i] && a[i] < d[j])
                d[j] = a[i];
}
```

Відновити відповідь можна використовуючи два масиви: масив "динаміки" d[] і масив "предків" p[]. Підтримуючи ці два масиви по ходу обчислення динаміки, в кінці буде неважко відновити шукану підпослідовність.

Завдання 3.14. *(приклад рішення)*

Задача 2012 на e-olymp.com.

Дана послідовність з n натуральних чисел. Знайдіть довжину її максимальної підпослідовності з елементів, які йдуть один за іншим, такий, що кожен елемент цієї підпослідовності на одиницю більший за попередній.

Вхідні дані

У першому рядку записано кількість n ($1 \le n \le 105$) елементів послідовності. У наступному рядку записана послідовність з n цілих чисел ai ($1 \le ai \le 106$), розділених пробілами.

Вихідні дані

Вивести довжину максимальної підпослідовності з елементів, які йдуть один за одним, кожен з яких на одиницю більший за попередній.

Тестовий приклад

Вхідні дані	Вихідні дані
6 1 2 4 3 4 5	3

```
#include <iostream>
using namespace std;
int main() {
    long n, x0, x, k = 1, max = 1;
    cin >> n;
    cin >> x0;
    for (int i = 1; i < n; i++) {
        cin >> x;
        if (x – x0 == 1) k++;
        else {
            if (k > max) max = k;
            k = 1;
        }
        if (k > max) max = k;
        x0 = x;
    }
    cout << max << endl;
    return 0;
}
```

Завдання для самостійної роботи

За варіантами

Завдання 1

6. Напишіть програму, яка для заданого користувачем речення підраховує, скільки в ньому розділових знаків.

7. Знайти в тексті всі слова, що мають 5 символів.

8. Вилучивши всі пропуски з деякого тексту, обчислити його нову довжину.

9. Визначити довжину двох введених рядків. Якщо довжина жодного з них не перевищує 25, об’єднати їх.

10. Визначити кількість входжень деякого символу, який вводиться з клавіатури, у заданий рядок.

11. Визначити, чи є слово паліндромом, тобто чи читається воно однаково в обох напрямках.

12. Заданий рядок символів розділити на дві рівні частини та визначити, у який з них кількість розділових знаків менша.

13. Дано рядок символів. З’ясувати, де більше крапок – до середини або після середини рядка.

14. Подвоїти всі символи «о» в заданому тексті та визначити довжину утвореного тексту.

15. У заданому рядку символів замінити всі крапки знаками окликів, а коми – тире.

16. У заданому рядку символів визначити кількість символів до першої крапки і після неї. Вивести ту частину рядка, в якій більше символів. Вважати, що хоча б одна крапка в рядку є.

17. Дано рядок символів. Одержати всі символи, розташовані після першої двокрапки.

18. Заданий рядок символів довільно розділити на дві частини і визначити в якій з них кількість букв «а» більше. (Ділення задається випадковим числом, що залежить від довжини рядка).

19. У заданому рядку символів видалити всі крапки і подвоїти всі пробіли.

20. З заданого рядка символів видалити всі голосні літери.

21. У заданому рядку видалити всі цифрові символи.

22. У заданому рядку визначити, чи є в ньому символи, відмінні від літер і пробілу і скільки їх.

Завдання 2

1) У заданому рядку символів замінити всі пари однакових символів одним відповідним символом.

2) Подвоїти всі літери в заданому тексті, якщо його довжина не перевищує n. У противному разі всі символи з номерами, що перевищують n, відкинути.

3) Задано деякий текст, що містить слова, записані літерами латинського алфавіту. Визначити кількість слів, які починаються із заданої літери.

4) Виключити із заданого рядка символів всі групи символів, розташовані між «(» і «)». Дужки також виключити. Передбачається, що усередині кожної групи дужок немає інших дужок.

5) У деякому введеному за допомогою клавіатури тексті всі символи «т» та «о», якщо вони є, потроїти.

6) Перевірити, чи міститься подвоєння літер у рядку, введеному за допомогою клавіатури. Якщо міститься, вивести порядкові номери

елементів, з яких це подвоєння починається, якщо не міститься, повідомити про це користувача.

7) Дано речення. Напишіть програму, яка знаходить слова, в яких буква «а» присутня не менше двох разів.

8) Зашифрувати задане слово A довжиною n символів, замінивши значення кожного символу його кодом, помноженим на порядковий номер літери у слові.

9) У заданому рядку символів замінити всі букви «а» та «с» на словосполучення «ас».

10) Задано рядок A і символ s. Вилучити з рядку символи, розміщені до символу s.

11) В заданому рядку символів дописати у кінці кожного слова літеру «а».

12) Дані два рядки символів. Визначити в якому з них кількість слів більша.

13) У заданому рядку символів замінити всі пропуски тире, причому, якщо декілька пропусків йде підряд, то замінити їх одним тире.

14) Замінити в деякому тексті всі поєднання літер «ма» на «мол». Символ «а», що стоїть після інших символів, не змінювати.

15) Задано деяке слово. Розбити його на групи по три символи. Кожну з груп подвоїти.

16) Задано рядок. Знайти в ньому всі слова, що містять літеру a.

17) Задано рядок. Знайти в ньому найбільшу кількість цифр, що йдуть підряд.

Завдання 3

Рішення задач с сайту e-olymp.com [4]

494, 909, 2803, 2164, 2165

Питання для самоконтролю

1. Яка з наступних функцій додає один рядок в кінець іншого?

    1) stradd();

    2) add();

    3) append();

    4) strcat(s1, s2);

2. Яка з наступних функцій порівнює два рядки?

    1) cmp();

    2) stringcompare();

    3) compare();

    4) strcmp();

3. Вкажіть статичний рядок

    1) char string [100];

    2) "Статичний рядок"

    3) 'Статичний рядок'

    4) string s;

4. Вкажіть коректне визначення строкової змінної

    1) string [20] mystr;

    2) char mystr [20];

    3) string mystr;

    4) string mystr [20];

5. Яким символом завершується С-рядок?

    1) '\0'

    2) '.'

    3) ''

    4) 'n'

6. Рядок «Привіт світ» буде показана на екрані чи ні?

```
int main(int argc, char ** argv) {
        int array [33];
        if (& array [4] <& array [23]) {
            std :: cout << "Привіт світ" << std :: endl;
```

```
        }
        return 0;
}
```

1) так

2) синтаксична помилка

3) ні

7. Як створити об'єкт типу string?

1) string [10] mystr;

2) char mystr [10];

3) str ing str;

4) string str [];

8. Як дізнатися довжину рядка типу string?

1) str.size();

2) str.length();

3) len();

4) strlen(s);

9. Як дізнатися, чи порожній рядок?

1) if (str.size () == 0);

2) if (str.size());

3) if (str.empty() == false);

4) if (str.empty());

10. У якому рядку записана конкатенація рядків типу string?

1) concat(str1, str2);

2) str = str1 + str2;

3) str1.append(str2);

4) strcat(str1, str2);

11. Як порівняти значення двох рядків?

1) if (str1 == str2);

2) compare strings;

3) if (str1 = str2);

4) if (str1! = str2);

12. Як перевернути рядок

1) reverse_iterator {str.r begin ()};

2) revers(str1, str2);

3) s.swap(str);

4) swap(s0, s1);

13. Як знайти індекс даного символу в рядку?

1) index = str.find('@');

2) ind = str.compare(ch);

3) char ch = str.at(0);

4) s.find(str, n);

14. Як дізнатися значення конкретного символу рядка, знаючи його порядковий номер?

1) s.find(n);

2) s.at(1);

3) int(c);

4) chr(n);

15. Звернення до символу рядка за його номером

1) str[3];

2) str.at(7);

3) s.find(n);

4) chr(n);

16. Як вставити в рядок символ

1) s.insert (pos, str, begin, count);

2) str.replace(n, m, str2);

3) s.push_back(symbol);

4) s.compare(str);

17. Як замінити символ в рядку або частині рядка?

1) str.replace(n, kol, str2);

2) s.clear();

3) s.compare(str);

4) s.erase(begin, count);

18. Вкажіть метод видалення даного символу з рядка.

1) s.erase(begin, count);

2) s.clear(c);

3) str.replace(n, m, str2);

4) chr(n);

19. Який метод дозволяє виділити підрядок в рядку?

1) s = str.substr(pos, n);

2) str = to_string(s);

3) stostr(str);

4) s.append(str);

20. Як перетворити рядок в число?

1) stoi(str);

2) to_string(n);

3) str.substr(p, n);

4) s.push(str);

21. Який метод викликається для перетворення числа в рядок?

1) to_string(c);

2) stostr(str);

3) s.append(str);

4) s.assign(str);

Література до розділу

6. Документація по C ++. Клас string . [Електронний ресурс]. - Режим доступу: http://www.cplusplus.com/reference/string/basic_string/

7. Підручник по C ++ . [Електронний ресурс]. - Режим доступу: http://cppstudio.com/

8. Рядки . [Електронний ресурс]. - Режим доступу: http://cppstudio.com/cat/309/325/

9. Сайт для программистов e-olymp. [Електронний ресурс]. - Режим доступу: e-olymp.com.

10. Сайт для программистов Timus Online Judge. [Електронний ресурс]. - Режим доступу: acm.timus.ru.

11. Таблица ASCII [Електронний ресурс]. - Режим доступу: https://snipp.ru/handbk/table-ascii

12. Уроки программирования на языке C++. [Електронний ресурс]. - Режим доступу: ravesli.com/uroki-cpp.

13. Учебник по C++. [Електронний ресурс]. – Режим доступу: http://cppstudio.com/cat/274/

14. Функції бібліотеки string.h . [Електронний ресурс]. - Режим доступу: http://cppworld.h16.ru/stdc/string_h.htm

15. Функція getline. [Електронний ресурс]. - Режим доступу: https://proginfo.ru/getline/

16. Шаблонний строковий клас string. [Електронний ресурс]. - Режим доступу: http://cppstudio.com/post/6110/

Розділ 4. Функції

4.1. Функції. Параметри функцій. Оператор повернення return

Теоретичні відомості

Функція - логічно завершений фрагмент програми, що має ім'я.

Функції дозволяють розділити великі обчислювальні задачі на підзадачі. Функції в С ++ бувають вбудовані (стандартні) і програмовані користувачем.

Функція має заголовок і тіло функції.

<Тип> <ім'я> (<тип> <ім'я параметра>,...)

{

<Тіло функції>

}

Виклик функції призводить до виконання деяких дій. Наприклад, при зверненні до функції printf() здійснюється виведення даних на екран. Якщо функція не повертає ніякого значення, вона має тип void. Функція може бути без параметрів або мати параметри.

```
void print() {
    cout << "Виводимо деякий текст";
}
int main() {
    print();
}
```

Параметри в оголошенні функції називаються формальними. При виконанні функції параметр може мати іншу назву. Але типи параметрів

повинні бути аналогічні. При зверненні до функції передаються фактичні параметри.

Якщо функція виконує деякі обчислювальні операції, то вказується тип значення, яке повертається. *Оператор return* дозволяє закінчити поточну функцію і повернути значення, яке було обчислено.

```
int sum(int a, int b) {              // a, b - формальні параметри
        return a + b;
}
int main() {
        int x, y;
        cin >> x >> y;
        cout << sum (x, y) << endl;
                // sum(x, y) - виклик функції; x, y - фактичні параметри
}
```

У момент виклику функції sum(x, y) відбуваються присвоювання a=x; b=y;

Так реалізується **передача параметрів за значенням**.

Функція може бути описана після головного методу main()

```
int sum (int, int);                  // прототип функції
int main () {
        int x, y;
        cin >> x >> y;
        cout << sum (x, y) << endl;
}
int sum (int a, int b) {             // функція
        return a + b;
}
```

Область видимості, локальні і глобальні змінні.

Приклади.

x, y – доступні всередині функції main()

a, b – доступні всередині функції sum(int a, int b)

i – доступна всередині циклу for(int i = 0; i <n; i ++)

Глобальні змінні доступні у всіх частинах програми.

Завдання 4.1. *(приклад рішення)*

Знайти суму елементів лінійного масиву, кратних 5.

```
#include <iostream>
using namespace std;
int mas[10];          // глобальне оголошення масиву
int summa() {         //  функція без параметрів
        int s = 0;
        for(int i = 0; i <10; i++) {
                if (mas[i] % 5 == 0)
                s + = mas[i];
        }
        return s;
}
int main() {
        for(int i = 0; i <10; i ++) {
                cin >> mas[i];
        }
        cout << summa() << endl;
}
```

Завдання 4.2. *(приклад рішення)*

Написати функцію, яка перевіряє, чи є підрядок short префіксом рядка longer

```
bool isPrefix (string short, string longer) {
    return (longer.substr (0, short.length ()) == short);
}
```

4.2. Способи передачі параметрів

Теоретичні відомості

Аргументи функцій можуть передаватися за значенням, за адресою або за посиланням.

Використовуйте:

передачу за значенням для основних типів даних;

передачу за адресою для вказівників або масивів;

передачу за (константним) посиланням для структур, класів, в тих випадках, коли потрібно, щоб функція змінювала значення аргументу або якщо функція повинна повертати більше одного результату.

Приклад передачі параметрів за значенням ми вже розглядали.

При передачі параметра за значенням x = b; і далі в підпрограмі йде обробка змінної x. На b це ніяк не віддзеркалюється.

```
void f(int x) {
    x = x * x;
    // x змінюється всередині f
}
int main() {
```

```
    int b = 5;
    f(b);
    cout << b << endl;          // 5
}
```

При передачі параметра за посиланням формальний параметр x стає синонімом фактичного параметра b. Тому при зміні x буде одночасно змінюватися і b.

```
void f(int & x) {
    x = x * x;
    // x змінюється всередині f
}
int main() {
    int b = 5;
    f(b);
    cout << b << endl; // 25
}
```

Розглянемо приклад, коли функція повинна повертати два значення.

```
void f(int a, int & res1, int & res2) {
    //. . . обчислення
    res1 =
    res2 =
}
int main() {
    int res_1, res_2;
```

```
        f(10, res_1, res_2);
        cout << « Результати :» << res_1 << " << res_2 << endl;
}
```

res1, res2 передані за посиланням для того, щоб в них збереглися результати обчислень і ми отримали вихідні дані.

Вказівники на функції дозволяють передати одну функцію як аргумент іншої функції.

Завдання 4.3. *(приклад рішення)*

Приклад функції, що повертає значення типу string на прикладі перевірки пароля.

```
#include <iostream>
#include <string>
using namespace std;
string check_pass(string password) {
        string valid_pass = "qwerty123";
        string message;
        if (password == valid_pass) {
                message = " Доступ дозволено ";
        } else {
                message = " Невірний пароль !";
        }
        return message;
}
int main() {
        string user_pass;
```

```
    cout << " Введіть пароль :";
    getline(cin, user_pass);
    cout << check_pass(user_pass) << endl;
    return 0;
}
```

В даному випадку функція **check_pass** має тип **string** , отже вона буде повертати рядок. Точка старту програми - функція **main()** , яка має бути присутня в кожній програмі.

Завдання 4.4. *(приклад рішення)*

Приклад функції, що повертає значення типу bool

```
#include <iostream>
#include <string>
using namespace std;
bool check_pass(string password) {
    string valid_pass = "qwerty123";
    if (valid_pass == password)
        return true;
    else
        return false;
}
int main() {
    string user_pass;
    cout << " Введіть пароль :";
    getline(cin, user_pass);
    if (check_pass (user_pass)) cout << " Доступ дозволено " << endl;
    else cout << "Неправильний пароль!" << endl;
```

```
    return 0;
}
```

Можна організувати повторне введення пароля за допомогою рекурсії (докладніше - нижче). Рекурсія - це функція, яка викликає сама себе.

Завдання 4.5. *(приклад рішення)*

```
#include <iostream>
#include <string>
using namespace std;
void get_pass() {
    string user_pass;
    cout << " Введіть пароль :";
    getline(cin, user_pass);
    if (! password_is_valid (user_pass)) {
        cout << "Неправильний пароль!" << endl;
        get_pass(); // знову звертаємося до get_pass()
    }
    else {
        cout << "Доступ дозволено" << endl;
    }
}
int main() {
    get_pass();
    return 0;
}
```

Функції полегшують роботу програмісту і набагато підвищують читаність і зрозумілість коду, дозволяють структурувати код і представити програму у вигляді послідовності підзадач.

4.3. Передача масивів у функцію

Теоретичні відомості

При передачі масиву в функцію в якості аргументу передається не сам масив, а вказівник на перший елемент масиву. Це реалізовано в прикладах 1), 2), 3). Третій є найбільш поширений варіант передачі масиву в функцію.

1) int function (int mas[10]);

2) int function (int mas[], int n);

3) int function (int *mas, int n);

4) int function (int &a, int n) {

```
int * mas = & a;
```

}

Сигнатура методу (функції) – це ім'я методу плюс параметри, які слідують в заданому порядку. У сигнатуру методу не входять значення, які повертаються або кидають виключення, а також модифікатори.

Завдання 4.6. *(приклад рішення)*

Створити функцію, яка визначає, чи є в лінійному масиві рівно один позитивний елемент.

Вхідні дані

Кількість елементів масиву n, далі – лінійний масив на n елементів.

Вихідні дані

Вивести true або false.

```
bool funArray(int * mas, const int n) {
        int k = 0;
        for (int i = 0; i <n; i ++) {
                if (mas[i]> 0)
                k ++;
        }
        if (k == 1)
                return true;
        else
                return false;
}
```

Завдання 4.7. *(приклад рішення)*

В даному двовимірному масиві знайти суму елементів зазначеного рядка.

Вхідні дані

Два цілих числа n і m – розміри двовимірного масиву, k – номер рядка. Потім – сам масив.

Вихідні дані

Сума елементів k-го рядка.

```
#include <iostream>
using namespace std;

int **creation_array(const int nstr, const int nstb) {
        int **mass = new int *[nstr];
        for (int i = 0; i <nstr; i ++)
                mass[i] = new int [nstb];
        return mass;
}
```

```
void del(int **mass, const int nstr, const int nstb) {
        for (int i = 0; i <nstr; i ++)
                delete[] mass [i]; // видаляються рядки
        delete[] mass;
}
void init(int **mass, const int nstr, const int nstb) {
        for (int i = 0; i <nstr; i ++)
                for (int j = 0; j <nstb; j ++)
                        mass[i][j] = rand() % 10 - 2;
}
void print_array(int **mass, const int nstr, const int nstb) {
        for (int i = 0; i <nstr; i ++) {
                for (int j = 0; j <nstb; j ++)
                        cout << mass[i][j] << ";
                cout << endl;
        }
}
int summa(int *mas, int m) {
        int s = 0;
        for (int j = 0; j <m; j ++) {
                s + = mas[j];
        }
        return s;
}
int main() {
        int n, m, k;
        cin >> n >> m >> k;
        int **arr = creation_array (n, m);
        init(arr, n, m);
        print_array(arr, n, m);
```

```
        //в summa передається масив arr[k] і кількість елементів
        cout << summa(arr[k], m) << endl;
        del(arr, n, m);
}
```

4.4. Види пам'яті

Теоретичні відомості

Існує 3 види пам'яті: статична, автоматична (стек) і динамічна (купа).

Статична пам'ять виділяється до початку виконання програми. Така пам'ять доступна протягом усього часу виконання програми. Для розміщення об'єкта в статичної пам'яті досить задекларувати його в глобальному контексті або в main().

```
int id = 150; // визначення статичної глобальної змінної
int main() {
        std :: cout << id + 8; // її використання
}
```

Стек це область пам'яті, організована за принципом LIFO («останнім прийшов – першим пішов»). Іншими словами, додавати і видаляти значення в стеку можна тільки з однієї і тієї ж сторони. Розміщення на стеку є основний спосіб виділення пам'яті. На стеку автоматично розміщуються аргументи і локальні змінні функцій, інша метаінформація при виклику функцій. При виході із функції пам'ять очищується. Розмір пам'яті стека є фіксований, зазвичай – 1 мегабайт. Максимальний розмір залежить від конкретної обчислювальної системи і налаштувань компілятора або лінковщика.

Динамічна пам'ять виділяється з купи. Це область пам'яті операційної системи, яка виділяється на вимогу вказівки. Вона використовується для роботи з динамічними структурами. Об'єм пам'яті

для купи залежить від того, скільки пам'яті просить програма функцією malloc. При нестачі статичної пам'яті або стека використовується динамічна пам'ять.

Розглянемо приклади використання різних видів пам'яті при оголошенні масивів. Якщо масив оголошений статично, то він розміщується в статичній пам'яті. Під масив, оголошений локально, пам'ять виділяється на стеку. При цьому, треба враховувати обмежений розмір стека при виборі розміру локального масиву. Динамічні масиви розміщуються в динамічній пам'яті.

4.5. Рекурсивні функції.

Теоретичні відомості

Рекурсія означає повернення. Рекурсивної називається функція, що викликає сама себе. Рекурсивна функція використовує стек .

Глибина рекурсії – це кількість викликів.

Рекурсивний спуск – це заповнення стека викликами рекурсивної функції.

Рекурсивний підйом – повернення значень і очищення стека.

Задача 4.8. *(приклад рішення)*

Обчислити факторіал натурального числа, використовуючи рекурсивну функцію.

```
#include <iostream>

using namespace std;

int factorial(int);
```

```
int main() {
        int a = 3;
        int result = factorial(a);
        cout << result << endl;
}
int factorial(int n) {
        if (n == 0)  return 1;
        return n * factorial(n - 1);
}
```

Стек викликів заповнюється з вершини і буде виглядати наступним чином:

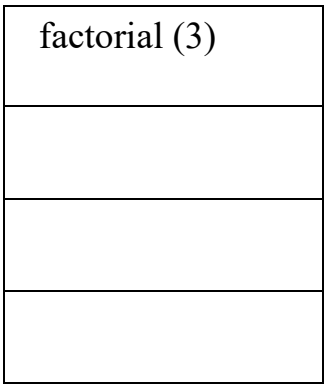


Рис.4.1. Перший виклик функції з параметром 3.

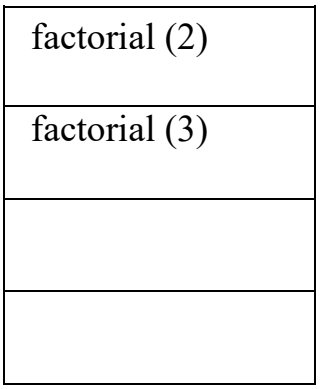
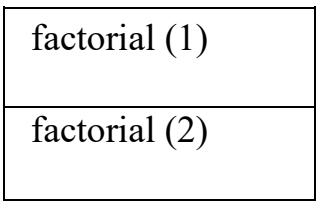


Рис.4.2. Другий виклик функції з параметром 2.



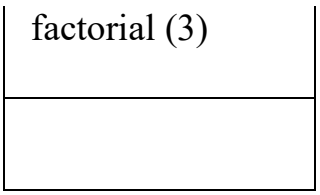


Рис.4.3. Третій виклик функції з параметром 1.

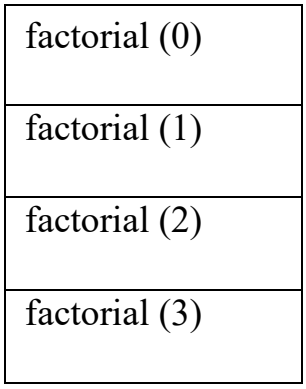


Рис.4.4. Четвертий виклик функції з параметром 0.

Повернення значень.

Перше значення, що повертається з вершини стека: factorial (0) = 1;

Потім factorial (1) = 1 * factorial(0) = 1;

factorial (2) = 2 * factorial(1) = 2;

factorial (3) =3 * factorial(2) = 6;

Задача 4.9. *(приклад рішення)*

Фарбування лабіринту

Лабіринт являє собою квадрат, що складається з NxN сегментів. Кожен із сегментів може бути або порожнім, або заповненим монолітною кам'яною стіною. Гарантується, що лівий верхній і правий нижній сегменти порожні. Лабіринт обнесений зверху, знизу, зліва і справа стінами, що залишають вільними тільки лівий верхній і правий нижній кути. Директор лабіринту вирішив пофарбувати стіни лабіринту, видимі з середини (див. Рис.4.5). Допоможіть йому розрахувати кількість фарби, необхідної для цього.

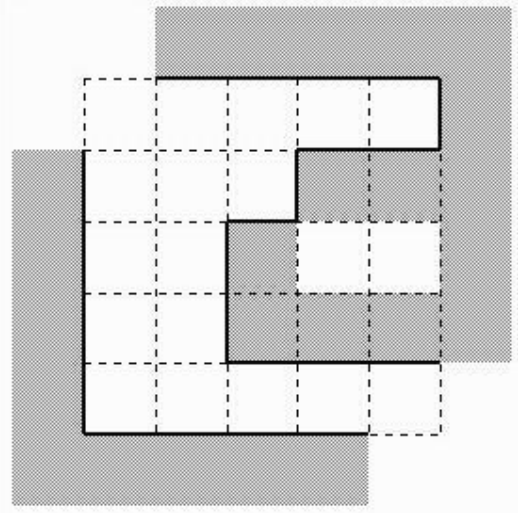


Рис.4.5. Лабіринт

Обмеження: 3 <= N <= 33, розмір сегмента 3 х 3 м, висота стін 3 м

Вхідні дані. У першому рядку знаходиться число N, потім йдуть N рядків по N символів: точка позначає порожній сегмент, решітка - сегмент зі стіною.

Вихідні дані. Вивести одне число - площу видимої частини внутрішніх стін лабіринту у квадратних метрах.

Тестовий приклад

Вхідні дані	Вихідні дані
5 ....## ..#.. ..###	198

```
#include<iostream>
using namespace std;
char **arr;
void paint(int i, int j) {
   if (arr[i][j] == '.') {
      arr[i][j] = '$';
      paint(i + 1, j);
      paint(i – 1, j);
      paint(i, j + 1);
      paint(i, j – 1);
   }
}
int main() {
   int n, k = 0;
   cin >> n;
   arr = new char *[n + 2];
   for (int i = 0; i < n + 2; i++) {
      arr[i] = new char [n + 2];
   }
   for (int i = 0; i < n + 2; i++) {
      for (int j = 0; j < n + 2; j++) {
         arr[i][j] = '#';
      }
   }
```

```
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            cin >> arr[i][j];
        }
    }
    paint(1, 1);
    if (arr[n][n] == '.') paint(n, n);
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            if (arr[i][j] == '$') {
                if (arr[i + 1][j] == '#') k++;
                if (arr[i – 1][j] == '#') k++;
                if (arr[i][j + 1] == '#') k++;
                if (arr[i][j – 1] == '#') k++;
            }
        }
    }
    cout << 9 * (k – 4) << endl;
    return 0;
}
```

Завдання для самостійної роботи

Завдання 1

Вирішити завдання з сайту e-olymp.com

1289, 1312, 1326, 1684, 2205, 4716, 4717, 4718, 6277, 7365.

Завдання 2

За варіантами

1. Визначити функцію, яка повертає модуль заданого вектору на площині, якщо задані координати вектору x, y.

2. Визначити функцію, яка повертає площу трикутника з заданими вершинами на площині.

3. Визначити функцію, яка повертає відстань між двома точками на площині, якщо задані координати точок x1, y1, x2, y2.

4. Визначити функцію, яка повертає скалярний добуток двох векторів, які задані координатами x1, y1, x2, y2 на площині.

5. Обчислити скалярний добуток двох векторів, завданих користувачем, використовуючи відповідну функцію.

6. Визначити функцію, яка повертає суму цифр натурального числа.

7. Визначити функцію, яка повертає натуральне число, якщо аргумент функції є квадратом цього числа и нуль в іншому випадку.

8. Знайдіть теософську суму числа, введеного користувачем, використовуючи відповідну функцію (обчислювати суму цифр числа, до тих пір, поки не залишиться одна цифра).

9. Показати, що будь-яке число, при відніманні від нього суми його цифр ділиться на 9 без залишку.

10. Визначити функцію, яка повертає ***True*** , якщо аргумент є простим числом и ***False*** в іншому випадку.

11. Вивести всі прості числа від N1 до N2, використовуючи відповідну функцію.

12. Визначити функцію, яка повертає суму перших k непарних натуральних чисел.

13. Використовуючи відповідну функцію показати, що сума перших k непарних натуральних чисел дорівнює квадрату їх кількості.

14. Складіть програму, яка знаходить всі натуральні числа, менші ніж ***N*** , для яких виконується співвідношення ***a + b = c*** .

15. Визначити функцію, яка повертає із завданою точністю суму членів ряду 1/n.

16. Визначити функцію, яка повертає факторіал натурального числа ***n! = 1 * 2 * 3 * ... * n***

17. Обчислити факторіали чисел від ***N1*** до ***N2***, використовуючи відповідну функцію.

18. Визначити функцію, яка повертає парний факторіал натурального числа ***n !! = 2 * 4 * 6 ... * n.***

19. Обчислити парні факторіали чисел від ***N1*** до ***N2***, використовуючи відповідну функцію.

20. Визначити функцію, яка повертає непарний факторіал натурального числа ***n !!! = 1 * 3 * 5 ... * n.***

21. Обчислити непарні факторіали чисел від ***N1*** до ***N2***, використовуючи відповідну функцію.

22. Визначити функцію, яка повертає найбільший спільний дільник ***n*** и ***m*** .

23. Знайдіть найбільший спільний дільник двох чисел в діапазоні від ***N1*** до ***N2***, заданого користувачем, використовуючи відповідну функцію.

24. Знайдіть всі можливі цілі значення довжин сторін прямокутного трикутника в діапазоні від 1 до N, використовуючи відповідну функцію.

25. Обчислити факторіали чисел від ***N1*** до ***N2*** , використовуючи відповідну функцію.

26. Визначити, чи належать три точки, які задані користувачем, одній прямій, використовуючи відповідну функцію.

27. Визначте функцію, яка повертає площу трикутника з заданими вершинами на площині, використовуючи формулу Герона.

28. Визначте функцію, яка повертає напівпериметр трикутника, якщо задані координати его вершин.

29. Визначте функцію, яка повертає довжини сторін трикутника, якщо задані координати его вершин.

30. Обчислити площу трикутника з заданими користувачем вершинами, використовуючи відповідну функцію.

31. Визначте функцію, яка повертає ***True***, якщо точка ***D*** належить трикутнику ***ABC*** и ***False*** в іншому випадку.

32. Складіть програму, яка визначає, чи належить задана користувачем точка заданому трикутнику.

33. Визначити функцію, яка повертає кут між вектором и позитивним напрямом осі OX на площині. Обчислити цей кут, використовуючи відповідну функцію.

34. Визначити функцію, яка повертає кут між двома векторами на площині через координати векторів.

Питання для самоконтролю

1. Що таке функція?

1) блок коду, який можна використовувати обмежену кількість разів

2) це звичайний оператор

3) блок коду, який можна використовувати необмежену кількість разів

4) бібліотека

2. Які види підпрограм реалізують сучасні мови програмування?

1) модулі

2) функції

3) процедури

4) оператори

3. Що таке прототип функції?

1) так по-іншому називають функцію

2) це функція, але з відсутністю блоку коду

3) це відключення функцій при їх частому використанні

4. Що таке аргумент функції?

1) це змінна, в яку можна передати значення

2) це локальна змінна, яка знаходиться в функції

3) це параметр, який використовується при виконанні функції

4) це змінна argument

6. Що включає повне ім'я функції?

1) сигнатуру, ім'я функції

2) ім'я функції, тип значення, тип і імена аргументів

3) ім'я функції, локальні змінні, що знаходяться в функції

4) ім'я функції, число аргументів, типи аргументів

7. Як викликаються процедури і функції

1) по імені, як окремий оператор

2) в блоці опису

3) по імені в здійснених операторах

8. Вкажіть зарезервоване ключове слово для динамічного виділення пам'яті

1) create
2) malloc
3) value
4) new

Література до розділу

1. Документація по C ++. [Електронний ресурс]. - Режим доступу: http://www.cplusplus.com/reference/set/set/

2. Лекції по C ++. [Електронний ресурс]. - Режим доступу: http://cpp.6te.net/index.php/lessons

3. Математичні функції math.h. [Електронний ресурс]. - Режим доступу: http://cppworld.h16.ru/stdc/math.htm

4. Уроки по C ++. [Електронний ресурс]. - Режим доступу: http://www.youtube.com/watch?v=dhB-yPWdeIQ

5. Уроки C ++. [Електронний ресурс]. - Режим доступу: https://code-live.ru/tag/cpp-manual/

6. Підручник по C ++. [Електронний ресурс]. - Режим доступу: http://cppstudio.com/cat/274/

7. Підручник по C ++. [Електронний ресурс]. - Режим доступу: http://www.programmersclub.ru/main/

8. Функції бібліотеки stdlib.h. [Електронний ресурс]. - Режим доступу: http://cppworld.h16.ru/stdc/mem.htm

9. Функції із змінним списком параметрів. [Електронний ресурс]. - Режим доступу: http://www.cyberguru.ru/programming/cpp/cpp-velvet-way-page76.html.

Розділ 5. Базові алгоритми та їх складність

5.1. Алгоритми довгої арифметики

Теоретичні відомості

В усіх мовах програмування цілі типи обмежуються певною розрядністю. Наприклад, в C ++ максимальним цілим типом є long long, який може зберігати числа до 20 знаків ($a*10^{19}$).

У деяких мовах програмування, наприклад, в мові Java, арифметика великих чисел є вбудованою і реалізована класом BigInteger. Арифметика великих чисел є важливим поняттям в програмуванні. Тому визначимо спочатку поняття великого числа.

Велике число - це число, яке не ”вміщається” в базовий цілий тип. Таке число необхідно розмістити в пам'яті комп'ютера. При цьому використовують або масиви, в яких кожен елемент містить одну цифру великого числа, або рядки, як послідовності знаків довгих чисел.

Можна записувати числа в системі з основою (100, 1000 та ін.).

Велике число можна також зберігати у вигляді структури.

```
int max_size = 3000;      // розмір масиву залежить від умови задачі
int base = 100;           //основа системи числення
```

```
struct BigInteger {
    int count;                  //кількість цифр числа
    int digits [max_size];      //масив цифр в зворотному порядку
};
```

Завдання 5.1. *(приклад рішення)*

Задача 2618 на e-olymp.com.

Дано число **n** . Виведіть число **n** + **1**.

Вхідні дані: Дано невід'ємне ціле число **n**, кількість його цифр не перевищує 1 млн.

Вихідні дані: Виведіть число **n** + **1**.

Тестовий приклад

Вхідні дані	Вихідні дані
5	6

```
#include <iostream>
using namespace std;
int main() {
    string s;
    bool tens = false;
    cin >> s;
    for (int i = s.length() - 1; i > = 0; i--) {
        if (s[i] == '9' &&! tens) {
            s[i] = '0';
```

```
            }
            else {
                s[i] + = 1;
                tens = true;
                break;
            }
        }
        if (! tens)
            cout << '1' << s << endl;
        else
            cout << s << endl;
}
```

Завдання 5.2. *(алгоритм)*

Задача 271 на e-olymp.com.

Знайти значення факторіала цілого числа **n**.

Вхідні дані: Одне ціле число **n** (**0** ≤ **n** ≤ **3000**).

Вихідні дані: Факторіал числа **n**.

Тестовий приклад

Вхідні дані	Вихідні дані
5	120

У задачі програма обчислює 3000! У цьому числі 9131 знак, на кінці – 748 нулів. Стандартний алгоритм не дає потрібного рішення, оскільки

найбільше ціле число, яке можна помістити в змінну типу long long має всього лише 20 знаків. Оголосимо масив на 10 000 елементів, де будемо розміщувати результат. Процес обчислення факторіала повторює алгоритм множення в стовпчик.

```
#include <iostream>

using namespace std;

int main() {

	int n, res, carry = 0, start, k = 9132;

	bool flag = false;

	cin >> n;

	int * arr = new int [k];

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

		arr[i] = 0;

	}

	arr[k - 1] = 1;

	for (int i = 1; i <= n; i ++) {		// наступне натуральне число

		for (int j = k - 1; j > = 0; j--) {

			res = arr[j] * i + carry;

			if (res> 9) {

				arr[j] = res% 10;

				carry = res / 10;

			} else {

				arr[j] = res;
```

```
                    carry = 0;
                }
            }
        }
        int i = 0;
        while (i <k &&! flag) {
            if (arr[i] == 0)   i ++;
            else {
                start = i;
                flag = true;
            }
        }
        for (int j = start; j < k; j ++) {
            cout << arr[j];
        }
        cout << endl;
}
```

Завдання 5.3. *(самостійна робота)*

Задача 313 на e-olymp.com.

Знайти суму **2** -х натуральних чисел **A** і **B**, використовуючи алгоритми довгої арифметики.

Завдання 5.4. *(приклад рішення)*

Задача 5223 на e-olymp.com.

Пам'ять Ватсона досягла критичного стану. Це означає, що всі осередки його пам'яті заповнилися одиницями. Рибка дізналася, що якщо всю пам'ять Ватсона вважати одним великим шістнадцятковим числом, то це число буде ділитися на 7. Але не повірила і захотіла перевірити цей факт. Для цього Рибка дізналася у Ватсона, скільки ж осередків в його пам'яті. Виявилося, що їх дуже багато - таких великих чисел Рибка ще не бачила. Допоможіть їй перевірити факт подільності пам'яті Ватсона.

Вхідні дані

Одне ціле невід'ємне число, представлене в десятковому вигляді - кількість осередків в пам'яті Ватсона. Це число містить не більше 100 цифр.

Вихідні дані

Вивести результат перевірки у вигляді одного слова: yes - якщо пам'ять Ватсона ділиться на 7 , no - пам'ять Ватсона не ділиться на 7.

Тестовий приклад

Вхідні дані	Вихідні дані
9	yes

Дане число може містити до 100 символів, тому ми можемо помістити його в змінну s типу string.

Зауважимо, що в шістнадцятковому коді запис 111 означає:

$1*16^2 + 1*16 + 1 = 256 + 16 + 1 = 273$. Це число ділиться на 7.

Будь-яка послідовність з трійок одиниць буде давати число, яке ділиться на 7. Наприклад, $111111 = 16^5 + 16^4 + 16^3 + 16^2 + 16 + 1 = 16^3*(256 + 16 + 1) + (256 + 16 + 1) = (256 + 16 + 1)*(16^3 + 1) = 273*(16^3 + 1)$.

Тому, якщо пам'ять Ватсона заповнена одними одиницями, нам достатньо порахувати кількість одиниць. Якщо це число ділиться на 3, то такий запис шістнадцятирічного числа при перетворенні в десяткову систему числення дає число, яке ділиться на 7.

```
#include <iostream>

using namespace std;

int main() {

	string s;

	int sum = 0;

	cin >> s;

	for (int i = 0; i < s.length (); i ++) {

		sum + = s [i] – '0';		// int('0') = 48

	}

	if (sum % 3 == 0)

		cout << "yes" << endl;

	else

		cout << "no" << endl;

}
```

5.2. Метод повного перебору

Повний перебор (complete search) це найпростіший метод вирішення задач. З точки зору математики є строгим методом доказів. На змаганнях зі спортивного програмування використання цього методу швидше за все призведе до перевищення ліміту часу (Time Limit Exceeded), однак, це хороша стратегія для задач з невеликим об'ємом вхідних даних.

Завдання 5.5. *(приклад рішення)*

Задача 128 на сайте e-olymp.com.

Підрахуйте кількість щасливих квитків, у яких сума перших трьох цифр дорівнює n. Щасливим називається квиток з шестизначним номером, у якого сума перших трьох цифр дорівнює сумі трьох останніх.

Вхідні дані

Одне натуральне число n ($n \leq 27$).

Вихідні дані

Вивести кількість шуканих щасливих квитків.

Тестовий приклад

Вхідні дані	Вихідні дані
1	9

Проаналізуємо тестовий приклад.

Сума цифр в першій половині квитка дорівнює 1.

Тоді перша частина номера квитка може бути такою:

100

010

001

Для кожного випадку, який розглядається, є три варіанти продовження номера. Всього отримаємо 9 варіантів.

Отже, алгоритм вирішення буде наступним: обчислюємо кількість варіантів утворення першої половини номера квитків, у яких сума цифр дорівнює даному числу n. На кожен з отриманих варіантів буде така ж кількість варіантів продовження номера.

```
#include <iostream>
using namespace std;
int main() {
   int n, ticket = 0;
```

```
    cin >> n;
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 10; j++) {
            for (int k = 0; k < 10; k++) {
                if (i + j + k == n) ticket++;
            }
        }
    }
    cout << ticket * ticket << endl;
    return 0;
}
```

5.3. Послідовний і бінарний пошук

Розглянемо два підходи до вирішення завдання пошуку заданого числа в відсортованому масиві.

Завдання 5.6. *(приклад рішення)*

У відсортованому масиві, що складається з різних цілих чисел, знайти індекс елемента, рівного x. Якщо такого числа немає, вивести -1.

1-й спосіб. Послідовний пошук (sequential Search)

Послідовно кожен елемент масиву порівнюємо з заданим елементом. Якщо знаходимо збіг, то виводимо номер елемента. Якщо такий елемент не знайдений, виводимо –1.

```
int sequentialSearch(int x, int a[], int n) {
    for (int i = 0; i < n; i++)
        if (a[i] == x) return i;
    return –1;
}
```

2-й спосіб. Бінарний пошук елемента в відсортованому масиві.

Бінарний пошук працює наступним чином:

Знаходимо номер центрального елементу масиву.

Якщо шуканий елемент менше центрального елемента, то пошук продовжуємо на лівій половині інтервалу, інакше - на правій половині.

Якщо центральний елемент дорівнює елементу, який ми шукаємо, то повертаємо індекс цього елемента.

Якщо перебрали весь масив і не знайшли шуканого значення, то повертаємо контрольне значення з висновком not found (або -1).

Оскільки в кожній ітерації ми можемо відкидати відразу половину масиву, то швидкість виконання цього алгоритму досить велика. Навіть з масивом в мільйон елементів для визначення того, чи існує конкретне значення в цьому масиві чи ні, буде потрібно не більше 20 ітерацій! Однак бінарний пошук працює тільки в відсортованому масиві.

Зміна масиву (наприклад, відкидання половини елементів масиву) є витратною операцією, тому зазвичай масив не змінюється. Замість цього використовується два цілочисельних значення (min і max) для зберігання індексів мінімальної та максимальної меж пошуку елемента в масиві.

Розглянемо приклад роботи цього алгоритму з масивом {4, 5, 7, 10, 11, 14, 19, 20, 25} і шуканим елементом 7.

Спочатку min = 0, max = 8, так як ми перебираємо весь масив (всього елементів 9, але індекс останнього елемента дорівнює 8).

Ітерація №1: Обчислюємо середнє значення між min (0) і max (8), що дорівнює 4. Елемент №4 має значення 11, яке більше нашого шуканого значення. Оскільки масив відсортований, то ми знаємо, що всі елементи, які знаходяться праворуч від індексу 4 (і індекс 4 теж) є більше нашого шуканого числа. Тому min залишаємо колишнім, а max змінюємо на 3.

Ітерація №2: Обчислюємо середнє значення між min(0) і max(3), що дорівнює 1. Елемент №1 має значення 5, яке менше нашого шуканого значення. Оскільки масив відсортований, то ми знаємо, що всі елементи, які знаходяться зліва від індексу 1 (і індекс 1 теж) - менше нашого шуканого числа. Отже, min змінюємо на 2, а max залишаємо колишнім.

Ітерація №3: Обчислюємо середнє значення між min(2) і max(3), що дорівнює 2. Елемент №2 має значення 7, яке є нашим шуканим значенням. Повертаємо елемент №2.

Позначимо:

// array - це масив, в якому ми проводимо пошук.

// target - це шукане значення.

// min - це індекс мінімальної межі масиву, в якому ми здійснюємо пошук.

// max - це індекс максимальної межі масиву, в якому ми здійснюємо пошук.

//Функція binarySearch() повинна повертати індекс шуканого значення, якщо він виявлений. В іншому випадку, повертаємо -1.

```
int binarySearch(int *array, int target, int min, int max);
int main() {
        int array[] = { 4, 7, 9, 13, 15, 19, 22, 24, 28, 33, 37, 41, 43, 47, 50 };
        std::cout << "Enter a number: ";
        int x;
        std::cin >> x;
        int index = binarySearch(array, x, 0, 14);
        if (array[index] == x)
```

```
            std::cout << "Good! Your value " << x << " is on position "<<
index << " in array!\n";
        else
            std::cout << "Fail! Your value " << x << " isn't in array!\n";
        return 0;
}
```

a) Ітеративна версія функції binarySearch().

```
int binarySearch(int *array, int target, int min, int max) {
    while (min <= max)  {
        int  midpoint  =  min  +  ((max-min)  /  2);  //такої  спосіб  обчислення
            середини масиву уникає ймовірність виникнення переповнення
        if (target < array[midpoint])   //шукаємо на лівій половині масиву
            max = midpoint;          //змінює межу max
        else  min = midpoint + 1        // змінює межу min
        return midpoint;
}
    return –1;
}
```

b) Рекурсивна реалізація функції binarySearch().

```
int binarySearch(int *array, int target, int min, int max) {
   if (min > max)  return -1;
   int midpoint = min + ((max-min) / 2);
   if (target < array[midpoint]) {
        return binarySearch(array, target, min, midpoint);
```

```
    }
    else {
        return binarySearch(array, target, midpoint + 1, max);
    }
}
```

Завдання 5.7. *(алгоритм)*

Швидке піднесення до степені.

Ідея алгоритму полягає в «швидкому» наближенні до результату, за O(log n) кроків.

Наприклад, потрібно обчислити 2^{64}. //a = 2, b = 64.

На кожному кроці виконуємо команди: a = a*a; b = b / 2;

Отримаємо:

a= 4; //2^2 b=32;

a=16; //2^4 b=16;

a=256; //2^8 b=8;

a=65536; //2^{16} b=4;

a=2^{32} b=2;

a=2^{64} b=1;

Таким чином, за 6 кроків ми отримали потрібний результат.

Перевірте роботу програми при a = 2, b = 64. Самостійно підготуйте тестовий приклад для випадку, коли показник ступеня на якомусь етапі є непарним числом.

```
#include <iostream>
using namespace std;

int main() {
    int a, b, res = 1;
    cin >> a >> b;
```

```
    while (b != 0) {
        if (b % 2 == 0) {
            b /= 2;
            a *= a;
        } else {
            b--;
            res *= a;
        }
    }
    cout << res << endl;
}
```

5.4. Складність алгоритму

Складність алгоритмів в теорії чисел прийнято вимірювати кількістю арифметичних операцій (додавання, віднімання, множення і ділення із залишком), необхідних для виконання всіх дій, запропонованих алгоритмом.

Час роботи числових алгоритмів зазвичай аналізується шляхом підрахунку арифметичних операцій. Алгоритми сортування та пошуку аналізуються шляхом підрахунку кількості порівнянь пар елементів.

Загальна кількість операцій пропорційна фактичному часу виконання алгоритму. Тому складність алгоритму оцінюють по часу, який потрібен для обробки дуже великого набору даних. Вона залежить від обсягу вхідних даних і, відповідно, кількості операцій над ними.

Складність алгоритму записується у вигляді O-нотації (від нім. «Ordnung» - порядок).

Приклади:

1. Дана лінійна матриця розміром n. Пошук максимального елемента має O(n) - лінійну складність.

2. Дана квадратна матриця nxn. Обчислення суми всіх елементів має O(n*n) квадратичну складність.

Завдання 5.8. *(приклад рішення)*

Розглянемо кілька рішень задачі «Чорна п'ятниця».

Завтра чорна п'ятниця – найбільший новорічний розпродаж. Степан, як господар магазину, прийняв рішення, що ціни усіх товарів буде знижено на 25%. Він з'ясував, що початкові ціни на усі товари ділились на 4, тому після зниження цін усі ціни також виражаються цілим числом. Ввечері перед розпродажем Степан зняв цінники з усіх товарів і надрукував для кожного товару ще один цінник зі знижкою. Він залишив їх на столі, щоб зранку їх розвісити. Але, коли він прийшов зранку в магазин, то виявилось, що прибиральниця змішала усі цінники разом, і тепер Степану потрібно відділити старі цінники від нових. Допоможіть йому.

Вхідні дані:

Перший рядок вхідного файлу містить одне число ***N (2 ≤ N ≤ 105)***, ***N*** - парне. Наступні ***N*** рядків містять додатні числа не більші за 109, які йдуть в порядку неспадання по одному в рядку - числа записані на усіх цінниках (як старих, так і нових). Гарантується, що вхідні дані коректні, тобто рішення існує.

Вихідні дані:

Виведіть ***N/2*** цілих чисел в порядку неспадання - вартості товарів після зниження цін.

Рішення 1. Складність O(n2)

Беремо першу за порядком картку arr[i] і перевіряємо, чи є для неї «старий» цінник arr [j], тобто arr [i] * 4/3 == arr [j].

Нову картку виводимо: cout << arr [i] << endl; а «стару» – видаляємо arr[j] = 0;

#include <iostream>

```
using namespace std;
int main() {
	int n;
	cin >> n;
	int arr[n];
	for (int i = 0; i < n; i++) {
		cin >> arr[i];
	}
	for (int i = 0; i < n - 1; i++) {
		for (int j = i + 1; j < n; j++) {
			if ((arr[i]*4 / 3 == arr[j]) && arr[j] != 0) {
				cout<<arr[i]<<endl;
				arr[j] = 0;
				break;
			}
		}
	}
}
```

Рішення 2. Використовуємо двійковий пошук. складність O(n*log n)

```
	#include <iostream>
	using namespace std;
	int n;
	int *arr;
	int search(int q) {
		int m, l = 0, r = n - 1;
		while (r > l) {
			m = (r + l) / 2;
			if (q > arr[m])
				l = m + 1;
```

```
                else r = m;
            }
            return r;
        }
        int main() {
         int q, ind;
         cin >> n;
         arr = new int [n];
         for (int i = 0; i < n; i++) {
                cin >> arr[i];
        }
         for (int i = 0; i < n - 1; i++) {
                if (arr[i] != 0) {
                        q = arr[i]*4 / 3;
                        ind = search(q);
                        cout << arr[i] << endl;
                        arr[ind] = 0;
                }
        }
}
```

Рішення 3. Складність O(n)

Проходимо по масиву вхідних даних один раз. Розкладаємо відповідну пару карток в масиви: нові ціни та старі ціни.

```
#include <iostream>
using namespace std;
int main() {
   int n, q;
   cin >> n;
   int prices1[n],  prices2[n];     //  нові ціни та старі ціни
   int j1 = 0,  j2 = 0, k2 = 0;
```

```
    for (int i = 0; i < n; i++) {
            cin >> q;
            if (prices2[k2]  != q) {          //якщо в «старих» цінах немає q
                    prices1[j1++] = q;
                    prices2[j2++] = q * 4 / 3;
            }
            else k2++;
    }
    for (int i = 0; i < n/2; i++) {
            cout << prices1[i] << endl;
    }
}
```

5.5. Методи динамічного програмування

Динамічне програмування - метод вирішення завдань шляхом їх розкладання на частини, невеликі та менш складні завдання. Рішення задач методами динамічного програмування проводиться на основі сформульованого Р. Е. Беллманом принципу оптимальності: оптимальна поведінка має ту властивість, що яким би не був первісний стан системи і початкове рішення, подальше рішення має визначати оптимальну поведінку відносно стану, отриманого в результаті початкового рішення. З цього випливає, що планування кожного кроку має проводитися з урахуванням отримання оптимального кінцевого результату.

Динамічне програмування «зверху» - це просте запам'ятовування результатів рішення підзадач, які можуть повторно використовуватися в наступних кроках рішення.

Динамічне програмування «знизу» включає в себе переформулювання складного завдання у вигляді рекурсивної послідовності простіших підзадач.

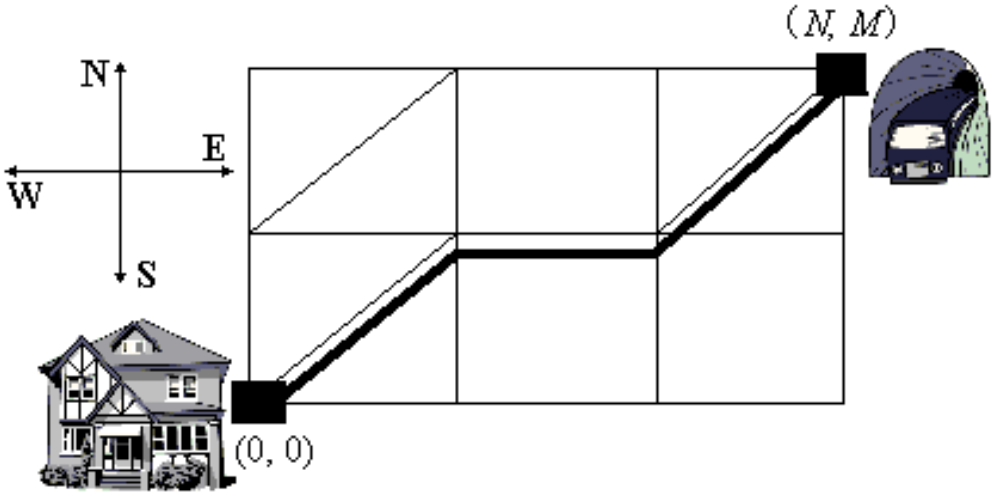
Завдання 5.9. *(приклад рішення)*

Задача 1119 на acm.timus.ru

Метро.

Багато програмістів КБ люблять добиратися до роботи на метро. Благо, головний офіс розташований зовсім недалеко від станції. Ну а оскільки сидячий спосіб життя вимагає активних фізичних навантажень у вільний від роботи час, багато співробітників - в тому числі і Никифор - ходять від будинку до метро пішки.

Никифор живе в такому районі нашого міста, де вулиці утворюють правильну сітку кварталів; всі квартали є квадратами з довжиною сторони, дорівнює 100 метрам. Вхід на станцію метро розташований на одному з перехресть; Никифор починає свій шлях з іншого перехрестя, який розташований на південь і на захід від входу в метро. Природно, що вийшовши з дому, Никифор завжди йде по вулицях, які ведуть або на північ, або на схід. Деякі квартали, які зустрічаються йому на шляху, він може також перетнути по діагоналі, що веде з південно-західного кута кварталу в північно-східний. Таким чином, деякі з маршрутів (провідних завжди на північ, схід або північний схід), виявляються коротше інших. Никифора цікавить, скільки часу знадобиться йому на подолання найкоротшого маршруту; для цього йому потрібно знати його довжину.



Ви повинні написати програму, яка за наявною інформацією про вид сітки кварталів розраховує довжину найкоротшого маршруту з південно-західного кута в північно-східний.

Вхідні дані

У першому рядку знаходяться два цілих числа N і M (0 < N , M ≤ 1000 чоловік) - розмір сітки кварталів із заходу на схід і з півдня на північ відповідно. Никифор починає шлях з перехрестя, який знаходиться на північний захід від кварталу з координатами (1, 1); станція метро знаходиться на північний схід від кварталу з координатами (N , M). У другому рядку знаходиться ціле число K (0 ≤ K ≤ 100) - кількість кварталів, через які можна пройти навскіс. Далі йдуть K рядків з парами цілих позитивних чисел, розділених пробілами - координатами таких кварталів.

Вихідні дані

Потрібно вивести довжину найкоротшого шляху від будинку Никифора до станції метро в метрах, округлений до цілих метрів.

Тестовий приклад

Вхідні дані	Вихідні дані
3 2 3 1 1 3 2 1 2	383

```
#include <iostream>
#include <cmath>
using namespace std;
//Функція обчислює мінімальне з трьох чисел
```

```
double min3(double a, double b, double c);
int main() {
    int n, m, k, x, y, i, j;
    double d,ans;
    cin >> n >> m >> k;
    double arr[n + 1][m + 1];
    bool b[n + 1][m + 1];
    for (i = 0; i <= n; i++) {
            for (j = 0; j <= m; j++) {
                    b[i][j] = false;
                    arr[i][j] = 0;
            }
    }
    for (i = 0; i < k; i++) {
            cin >> x >> y;
            b[x][y] = true;
    }
    for (i = 0; i <= n; i++) {
            arr[i][0] = i;
    }
    for (j = 0; j <= m; j++) {
            arr[0][j] = j;
    }
    d = sqrt(2);
```

```
    double l1, l2, l3, min_dist;
    for (i = 1; i <= n; i++) {
            for (j = 1; j <= m; j++) {
                    l1 = arr[i - 1][j] + 1;
                    l2 = arr[i][j - 1] + 1;
                    l3 = 2147483647;
                    if (b[i][j])
                            l3 = arr[i - 1][j - 1] + d;
                    min_dist = min3(l1, l2, l3);
                    arr[i][j] = min_dist;
            }
    }
    ans = min_dist * 100;
    printf("%.0f\n", ans);
}
double min3(double a, double b, double c) {
    return min(min(a,b),c);
}
```

Завдання 5.10. *(приклад рішення)*

Задача 838 на сайті e-olymp.com.

Маршрут

У таблиці з **N** рядків і **N** стовпців клітини заповнені цифрами від **0** до **9**. Потрібно знайти такий шлях з клітки (**1**, **1**) в клітку (**N**, **N**), щоб сума

цифр в клітинах, через які він пролягає, була мінімальною; з будь-якої клітини ходити можна тільки вниз або вправо.

Вхідні дані

У першому рядку знаходиться число **N** (**2** ≤ **N** ≤ **250**). У наступних **N** рядках містяться по **N** чисел без пробілів.

Вихідні дані

Виводяться **N** рядків по **N** символів. Символ решітка показує, що маршрут проходить через цю клітку, а точка - що не проходить. Якщо шляхів з мінімальною сумою цифр декілька, вивести будь-який.

Тестовий приклад

Вхідні дані	Вихідні дані
3 943 216 091	#.. ### ..#

```
#include<iostream>
using namespace std;
int main() {
   int n;
   char a;
   cin >> n;
   int numbers[n][n], sums[n][n];
   char directions[n][n], res[n][n];
   for (int i = 0; i < n; i++) {
```

```
        for (int j = 0; j < n; j++) {
            cin >> a;
            numbers[i][j] = a - '0';
        }
    }
    sums[0][0] = numbers[0][0];
    directions[0][0] = 'i';
    for (int i = 1; i < n; i++) {
        sums[0][i] = sums[0][i - 1] + numbers[0][i];
        directions[0][i] = 'l';
        sums[i][0] = sums[i - 1][0] + numbers[i][0];
        directions[i][0] = 'u';
    }
    for (int i = 1; i < n; i++) {
        for (int j = 1; j < n; j++) {
            sums[i][j] = min(sums[i - 1][j], sums[i][j - 1]) + numbers[i][j];
            if (sums[i - 1][j] == min(sums[i - 1][j], sums[i][j - 1])) {
                directions[i][j] = 'u';
            } else directions[i][j] = 'l';
        }
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            res[i][j] = '.';
```

```
        }
    }
    int i = n - 1;
    int j = n - 1;
    while (directions[i][j] != 'i') {
        res[i][j] = '#';
        if (directions[i][j] == 'u') {
            i--;
        } else if (directions[i][j] == 'l') {
            j--;
        }
    }
    res[0][0] = '#';
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cout << res[i][j];
        }
        cout << endl;
    }
    return 0;
}
```

Завдання для самостійної роботи

Рішення задач на сайті e-olymp.com

№№ 17, 267, 271, 314, 480, 986, 988, 9016, 1228, 1786.

Питання для повторення

Повторити базові алгоритми.

Підготувати повідомлення по темі відповідно до варіанта.

23. Алгоритм обміну значеннями двох змінних

24. Алгоритм пошуку найбільшого (найменшого) з двох чисел

25. Алгоритм пошуку найбільшого (найменшого) з трьох чисел

26. Алгоритм пошуку зростаючої підпослідовності найбільшої довжини

27. Алгоритм пошуку суми членів ряду

28. Алгоритм Евкліда для знаходження НСД

29. Сортування лінійних масивів

30. Сортування бульбашками

31. Швидке сортування

32. Множення матриць

33. Рекурсивне обчислення факторіала натурального числа

34. Алгоритми довгої арифметики

35. Метод повного перебору

36. Послідовний і двійковий пошук

37. Швидке піднесення до степені

Література до розділу

5. Алгоритмы и структуры данных. [Електронний ресурс]. – Режим доступу: http://cppstudio.com/cat/293/

6. Алгоритмы. [Електронний ресурс]. - Режим доступу: e-maxx.ru/algo.

7. Ахо А., Хопкрофт Д., Ульман Д. Структуры данных и алгоритмы. М.: Вильямс. 2000.

8. Караванова Т.П. Інформатика. Методи побудови алгоритмів та їх аналіз. К.:Генеза. 2007. 216 с.

9. Кениг, Э. Эффективное программирование на C++. Практическое программирование на примерах. Т. 2 / Э. Кениг, Б.Э. Му. М.: Вильямс. 2016. 368 с.

10. Кормен Т.,Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. 2-е издание. М.: Вильямс. 2006.

11. Лекции по алгоритмизации и основам программирования на языке C++. [Електронний ресурс]. - Режим доступу: http://teacher.ucoz.net/index/lekcii_po_s/0-4

12. Макаров. Программирование и основы алгоритмизации. [Електронний ресурс]. – Режим доступу: http://window.edu.ru/resource/126/25126/files/nwpi223.pdf

13. Підручник по C ++ . [Електронний ресурс]. - Режим доступу: http://cppstudio.com/

14. Программирование на C++. [Електронний ресурс]. - Режим доступу: http://www.cplusplus.com/

15. Сайт для программистов. [Електронний ресурс]. - Режим доступу: http://www.e-olymp.com/

16. Страуступ, Б. Язык программирования C++. Специальное издание / Б. Страуступ. М.: Бином. 2015. 1136 с.

17. Топп, Уильям Форд. Структуры данных в C++: Пер. с англ. М.: ЗАО «Издательство БИНОМ». 2000. 816 с.

18. Хенкеманс, Д. Программирование на C++ / Д. Хенкеманс. М. Ли. СПб.: Символ-плюс. 2015. 416 с.

Термінологічний словник

Алгоритм. Набір інструкцій, які описують порядок дій виконавця, щоб досягти результату розв'язання задачі за скінченну кількість дій.

Аргумент. Ім'я змінної, що використовується в блоці оголошення функції для вказівки значення, яке передається в функцію в якості фактичного параметра.

Базові логічні операції. Операції: and, or, і not.

Булеви змінні. Логічні змінні, які можуть приймати одне з двох значень True і False.

Булевий вираз. Вираз, який містить логічні змінні, які з'єднані логічними операціями.

Виклик функції. Здійснюється за допомогою вказівки імені функції, за яким слідує перелік фактичних параметрів, укладених в дужки.

Декремент. Операція зменшення змінної цілого типу.

Диз'юнкція. Операція роз'єднання (or) над логічними змінними.

Заголовок. Перший рядок визначення функції.

Заперечення. Зміна значення логічної змінної на протилежне (not).

Ініціалізація. Присвоєння початкового значення змінної.

Інкремент. Операція збільшення змінної цілого типу (часто на одиницю).

Інструкція розгалуження, також «Умовний перехід». Конструкція мови програмування, яка дозволяє виконувати різні дії залежно від булевого значення умови.

Інструкція. Мінімальна програмна одиниця, яка описує логічно і алгоритмічно завершену дію, яку може виконати комп'ютер.

Ітерація. Один прохід при виконанні блоку інструкцій циклу.

Ключові слова. Зарезервовані буквосполучення, які використовуються і інтерпретуються компілятором в програмах завжди в одному і тому ж сенсі.

Команда присвоювання. Позначається знаком рівності і полягає в обчисленні виразу в правій частині і присвоєнні отриманого значення змінній з лівої частини.

Коментар. Інформація про програму, яка призначена для інших програмістів (які читають вихідний код) і яка не впливає на виконання програми.

Компілювання. Переклад програми, написаної мовою високого рівня, на мову обчислювальної машини для безпосереднього виконання.

Компонування, зв'язування, збірка або лінковка. Це останній етап процесу отримання виконуваного файлу, що складається зі зв'язаних воєдино об'єктних файлів проекту.

Комп'ютерна програма. Запис алгоритму розв'язання задачі мовою програмування.

Конкатенація. Операція склеювання строкових змінних.

Кон'юнкція. Логічне множення (and).

Локальна змінна. Змінна, яка визначена всередині функції і існує, поки виконується функція.

Масив. Складовий тип, який містить набір однотипних елементів іншого типу.

Мова низького рівня. Мова програмування, написана у формі інструкцій для комп'ютера. Також називається «машинною мовою», «в кодах» або «мовою асемблера».

Мова програмування високого рівня. Мова програмування, написана мовою, зрозумілою людям.

Модуль. Файл, який містить програмні ресурси у вигляді функцій, констант, класів.

Налагодження. Процес знаходження і виправлення помилок в програмі.

Нескінечний цикл. Цикл, який не може завершитися, оскільки умова закінчення циклу ніколи не виконується.

Об'єктний код. Програма на мові машинних кодів з частковим збереженням символьної інформації, необхідної в процесі лінковки.

Операнд. Аргумент операції.

Оператор. Спеціальний символ або група символів, які повідомляють інтерпретатор про необхідність виконання певних дій над операндами.

Операції порівняння. Логічні операції: ==, ! =, >, <, > =, <=

Операція. Сукупність дій над впорядкованою послідовністю чисел відповідно до набору правил.

Переносимість. Властивість програми, що полягає в можливості виконання її на різних типах обчислювальних машин та під керуванням різних операційних систем.

Помилка часу виконання (runtime_error). Загальний тип винятків, які виникають під час виконання програми. У більшості випадків причинами виключень є невірні вхідні дані.

Природна мова. Будь-яка з розмовних мов, якою розмовляють люди і яка розвивається природним шляхом.

Програма. Запис алгоритму розв'язання задачі мовою програмування.

Простір імен. Абстрактне сховище, створене для логічного групування унікальних ідентифікаторів.

Рядки (string). Тип даних, який представляє послідовності символів.

Семантична помилка. Смислова, логічна помилка.

Середовище розробки. Сукупність інструментів, які спрощують процес програмування та створення програм.

Синтаксис. Правила мови програмування високого рівня.

Синтаксична помилка. Помилка в тексті програми, яка порушує правила запису алгоритмічних конструкцій і тому робить неможливою переведення тексту програми компілятором в машинну форму.

Сортування. Упорядкування елементів в масиві або списку за зростанням або спаданням.

Структура циклу. Алгоритмічна структура, яка передбачає повторення послідовності дій при виконанні деякої умови.

Тип для чисел з плаваючою точкою (float або double). Типи даних, які представляють числа з дробовою частиною.

Тіло функції. Набір приписів, який визначає послідовність інструкцій, яка виконується після виклику функції.

Умови. Булеви вирази в інструкції розгалуження або циклі, які визначають, яка гілка інструкції виконується.

Фактичний параметр. Значення, що передається функції при її виклику. Значення присвоюється відповідному параметру функції.

Формальна мова. Будь-яка з мов, які розробляють люди для спеціальних цілей, таких, як математична мова або мова програмування.

Формальний параметр (аргумент). Змінна, що є частиною оголошення функції і значення якої встановлюється при виклику функції.

Функція print(). Функція, яка виводить інформацію на екран.

Функція. Запис алгоритму, який вирішує підзадачу і є складовою частиною програми.

Цикл з передумовою. Цикл, в якому умова продовження циклу перевіряється до першого виконання блоку команд циклу.

Цикл з постумовою. Цикл, в якому умова закінчення циклу перевіряється після першого виконання блоку команд циклу.

Цілий тип даних (int). Тип даних, який представляє цілі числа в інтервалі від -2 147 483 648 до 2 147 483 647.

Навчальне видання

О. С.Булгакова,
В. В. Зосімов,
Г. В. Ходякова

АЛГОРИТМІЗАЦІЯ І ПРОГРАМУВАННЯ: ТЕОРІЯ ТА ПРАКТИКА

Навчальний посібник

для дистанційного навчання

Формат 60×841/16. Ум. друк. арк. 8,0. Тираж 100 пр. Зам. № 684-657.

ВИГОТОВЛЮВАЧ
СПД Румянцева Г. В.
54038, м. Миколаїв, вул. Бузника, 5/1.