# 1 Поняття про алгоритмізацію. Поняття про алгоритм та його властивості. Типи команд. Типи алгоритмів

**Алгоритм** — це скінченна послідовність указівок на виконання дій, спрямованих на розв'язування задачі. ... У повсякденному житті людина стикається з **алгоритмами**, що визначають послідовність дій різної природи.

**Базові алгоритмічні структури, типи алгоритмів:**

1. слідування
2. розгалуження
3. повторення.

**Слідування** — це така форма організації вказівок в алгоритмі, за якої дії виконуються послідовно одна за одною, без пропусків або повторень.

**Розгалуження** — це така форма організації дій, коли залежно від виконання або невиконання певної умови виконується одна з двох вказівок

**Повторення (цикл)** — це така форма організації дій, за якої одна й та сама послідовність дій виконується кілька разів залежно від певної умови.

# 2 Способи запису алгоритмів. Мова блок-схем. Поняття про стандартні алгоритми та їх запис мовою блок-схем.

**Способи подання алгоритмів**

•           **Словесний** — подання алгоритмів, які при­значені на виконання людиною (наприклад, кулінарних рецептів, правил переходу ву­лиці тощо), природною мовою в усній або письмовій формі.

•           **Формульно-словесний**  —  подання  алгорит­мів у навчальній та науковій діяльності за допомогою мови математичних формул, хімічних процесів тощо зі словесними по­ясненнями природною мовою.

•           **Графічний** — подання алгоритмів у вигляді графічних схем (блок-схем або структурних схем) для спрощення розробки та аналізу  алгоритмів, полегшення переходу від запису алгоритмів до написання програм.

•           **Програмний** — подання алгоритмів мовою програмування для їх подальшого опрацювання на комп'ютері.

•           **Інші способи** — специфічні способи подання алгоритмів, такі, як запис музики за допо­могою нот тощо.

**Метод блок-схем**

**Блок-схема алгоритму** — це графічне зобра­ження алгоритму у вигляді спеціальних блоків із необхідними словесними поясненнями. Кож­ний етап алгоритму на блок-схемі подається у вигляді геометричної фігури, яка має певну форму залежно від характеру дії

**Блоки на схемі з'єднуються лініями зв'язку** (лініями потоку), які визначають послідовність виконання операцій та утворюють логічну структуру алгоритму.

# 3 Основні відомості про структуру програми та основні елементи МП С++.

Основними частинами типової структури програми на С++ є такі:

* директиви препроцесорної обробки;
* опис зовнішніх змінних (вихідних даних і результатів) та функцій;
* функції програми;
* головна функція — програми **main()**, що має вигляд

# 4 Поняття про директиви препроцесора #include, #define. Заголовочні файли.

Заголовки включаються в текст програми за допомогою директиви препроцесора #include. Директиви препроцесора починаються зі знака "дієз" (#), який повинен бути найпершим символом рядка. Програма, яка обробляє ці директиви, називається препроцесором (у сучасних компіляторах препроцесор зазвичай є частиною самого компілятора).

Директива #include включає в програму вміст зазначеного файлу. Файл може бути зазначено двома способами:

#include <some\_file.h>

#include "my\_file.h"

Якщо ім'я файлу укладено в кутові дужки (<>), вважається, що нам потрібен якийсь стандартний заголовний файл, і компілятор шукає цей файл в зумовлених місцях. (Спосіб визначення цих місць сильно розрізняється для різних платформ і реалізацій). Подвійні лапки означають, що заголовний файл - користувацький, і його пошук починається з того каталогу, де знаходиться вихідний текст програми.

Заголовний файл також може містити директиви #include. Тому іноді важко зрозуміти, які ж конкретно заголовні файли включені в даний вихідний текст, і деякі заголовки можуть виявитися включеними кілька разів. Уникнути цього дозволяють умовні директиви препроцесора. Розглянемо приклад:

# 5 Типи даних. Перетворення типів. Операція приведення

Обробка даних різного типу є головною метою будь-якої програми. Кожне з даних характеризується класом пам’яті, ім’ям, типом і значенням. Імена дозволяють ідентифікувати дані, тобто відрізняти їх між собою. Програміст обирає тип кожної величини, що використовується для подання реальних об’єктів. Тип задає множину можливих значень даних і способи їх зберігання, перетворення та використання.

**Обов’язкове оголошення типу даних дозволяє** компілятору робити перевірку допустимості різних конструкцій програми.

Усі типи даних мови C++ можна розділити на **основні** (базові) і **складені**. Основні типи визначені для представлення цілих, дійсних, символьних і логічних даних. На основі цих типів вводиться опис складених типів, до яких належать масиви, перелічення, функції, структури, посилання, покажчики, об’єднання і класи.

**Основні типи даних** (*див. табл. 3.1*) часто називають арифметичними, тому що їх можна використовувати в арифметичних операціях. Для опису основних типів мови C++ використовують такі службові слова:

* **int**(цілий);
* **char**(символьний);
* **bool**(логічний);
* **float**(дійсний);
* **double**(дійсний з подвійною точністю);
* **void**(порожній, не має значення).

Типи**int, char, bool**називають***цілими***, а типи **float** та **double** —***дійсними з плаваючою крапкою***. Код, що формує компілятор для обробки цілих величин, відрізняється від коду для величин з плаваючою крапкою.

Для уточнення внутрішнього подання та діапазону значень стандартних типів **мова C++ використовує чотири специфікатори типу**:

* **short**(короткий);
* **long**(довгий);
* **signed**(знаковий);
* **unsigned**(беззнаковий).

*Таблиця З.1*

Базові типи даних для ПК (платформа Intel)

|  |  |  |
| --- | --- | --- |
| Тип | Розмір, байт | Значення |
| **bool** | 1 | true або false |
| **unsigned short int** | 2 | від 0 до 65 535 |
| **short int** | 2 | від -32 768 до 32 767 |
| **unsigned long int** | 4 | від 0 до 4 294 967 295 |
| **long int** | 4 | від -2 147 483 648 до 2 147 483 647 |
| **int**(16 розрядів) | 2 | від -32 768 до 32 767 |
| **int**(32 розряди) | 4 | від -2 147 483 648 до 2 147 483 647 |
| **unsigned int** (16 розрядів) | 2 | від 0 до 65 535 |
| **unsigned int**(32 розряди) | 4 | від 0 до 4 294 967 295 |
| **char** | 1 | від 0 до 256 |
| **float** | 4 | від 1.2е-38 до 3.4е38 |
| **double** | 8 | від 2.2е-308 до 1.8е308 |
| **long double** | 10 | від 3.4е-4932 до 3.4е+4932 |

**Приведення (перетворення) типів** ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0" \o ") *type conversion*, *typecasting*, *coercion*) — в програмуванні це зміна типу сутності одного [типу даних](https://uk.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) в інший, що може відбуватися різними способами, явно чи неявно.

# 6 Введення- виведення в мові програмування С++. Функції puts, gets

У мові C++ дії, що пов’язані з операціями введення і виве­дення, виконуються за допомогою функцій бібліотек. Функції ведення і виведення бібліотек мови дозволяють читати дані з файлів та пристроїв і писати дані у файли і на пристрої.

Бібліотека мови C++ підтримує три рівня введення-виведення даних:

* введення-виведення потоку;
* введення-виведення нижнього рівня;
* введення-виведення для консолі і порту.

Функція puts () виводить рядок, зазначену str, на стандартний пристрій виводу. Нуль в кінці рядка транслюється в новий рядок.

Функція puts () повертає символ «новий рядок» в разі успіху і EOF - при невдачі

Функція gets () зчитує символи з stdin і поміщає їх в масив символів, на який вказує str. Символи зчитуються до тих, поки не зустрінеться новий рядок або EOF. Символ «новий рядок» не робиться частиною рядка, а транслюється в нульовий символ, що завершує рядок.

У разі успіху gets () повертає str, в іншому випадку вона повертає NULL. У разі помилки з читання вміст масиву, на який вказує str, не визначене.

Оскільки NULL повертається як в разі помилки, так і при досягненні кінця файлу, то слід використовувати feof () або ferror (), щоб з'ясувати, що саме сталося.

Кількість символів, що зчитуються gets (), не обмежується. Тому програміст повинен сам стежити за тим, щоб не вийти за межі масиву, на який вказує str.

# 7 Введення- виведення в мові програмування С++. Функції putchar, getchar

Макрос putcnar () записує символ, що знаходиться в молодшому байті ch, в файл stdout. Функці¬онально він еквівалентний putc (ch, sdout). Оскільки під час звернення до функції аргументи символьного типу приводяться до цілого типу, можна використовувати символьні змінні як аргументи putchar ().

У разі успіху putcahar () повертає записаний символ; в разі помилки буде повернутий EOF. Якщо вихідний потік відкритий в довічним режимі, то EOF є допустимим значенням для ch. Це означає, що доведеться використовувати ferror (), щоб з'ясувати, чи дійсно про¬ізошла помилка.

Макрос getchar () повертає черговий символ з файлу stdin. Символ зчитується як пере¬менная типу unsigned char, перетворена до цілого. При зчитуванні маркера кінця файлу повертається EOF.

Функціонально макрос getchar () еквівалентний getc (stdin).

# 8 Введення- виведення в мові програмування С++. Функції printf, scanf

Функція printf () записує в stdout аргументи зі списку arg-list під керуванням рядка, на яку вказує аргумент format.

Рядок, на яку вказує format, складається з об'єктів двох різних призначень. По-перше, це символи, які самі повинні бути виведені на екран. По-друге, це спеціфіка¬тори формату, що визначають вид, в якому будуть виведені аргументи зі списку arg-list. Спе¬ціфікатори формату складаються з символу відсоток, за яким слід код формату. Команди форматування наведені в таблиці. Кількість аргументів має точно соответство¬вать кількості специфікаторів формату, причому слідувати вони повинні в однаковому порядку.

|  |  |
| --- | --- |
| Код | Формат |
| %с | Символ типа char |
| %d | Десятичное число целого типа со знаком |
| %i | Десятичное число целого типа со знаком |
| %е | Научная нотация (е нижнего регистра) |
| %Е | Научная нотация (Е верхнего регистра) |
| %f | Десятичное число с плавающей точкой |
| %g | Использует код %е или %f — тот из них, который короче (при использовании %g используется е нижнего регистра) |
| %G | Использует код %Е или %f — тот из них, который короче (при использовании %G используется Е верхнего регистра) |
| %о | Восьмеричное целое число без знака |
| %s | Строка символов |
| %u | Десятичное число целого типа без знака |
| %х | Шестнадцатиричное целое число без знака (буквы нижнего регистра) |
| %Х | Шестнадцатиричное целое число без знака (буквы верхнего регистра) |
| %р | Выводит на экран значение указателя |
| %n | Ассоциированный аргумент — это указатель на переменную целого типа, в которую помещено количество символов, записанных на данный момент |
| %% | Выводит символ % |

Функция scanf() является процедурой ввода общего назначения, считывающей данные из пото­ка stdin. Она может считывать данные всех базовых типов и автоматически конвертировать их в нужный внутренний формат. Если бы printf() выполняла ввод, а не вывод, ее можно было бы назвать аналогом scanf().

Управляющая строка, на которую указывает format, состоит из символов трех типов:

* Спецификаторы формата
* Специальные символы
* Прочие символы (не специальные)

Спецификаторы формата следуют за символом процент и сообщают scanf(), данные какого типа будут считаны следующими. Коды спецификаторов приведены в таблице.

|  |  |
| --- | --- |
| Код | Значение |
| %с | Считать один символ |
| %d | Считать десятичное число целого типа |
| %i | Считать десятичное число целого типа |
| %е | Считать число с плавающей запятой |
| %f | Считать число с плавающей запятой |
| %g | Считать число с плавающей запятой |
| %о | Считать восьмеричное число |
| %s | Считать строку |
| %х | Считать шестнадцатиричное число |
| %р | Считать указатель |
| %n | Принимает целое значение, равное количеству считанных до текущего момента символов |
| %u | Считывает беззнаковое целое |
| %[] | Просматривает набор символов |
| %% | Считывает символ % |

# 9 Введення- виведення в мові програмування С++. Функції cin, cout

об'єкт std :: cout (який знаходиться в бібліотеці iostream) використовується для виведення даних на екран (в консольне вікно).

std :: cin є протилежністю std :: cout. У той час як std :: cout виводить дані в консоль за допомогою оператора виведення <<, std :: cin отримує дані від користувача за допомогою оператора введення >>. Використовуючи std :: cin ми можемо отримувати і обробляти користувальницький введення

# 10 Операції порівняння МП С++

|  |  |  |  |
| --- | --- | --- | --- |
| Оператор | Символ | Пример | Операция |
| Більше | > | x > y | true, если x больше y, в противном случае — false |
| Менше | < | x < y | true, если x меньше y, в противном случае — false |
| Більше або рівно | >= | x >= y | true, если x больше/равно y, в противном случае — false |
| Менше або рівно | <= | x <= y | true, если x меньше/равно y, в противном случае — false |
| Рівно | == | x == y | true, если x равно y, в противном случае — false |
| Не рівно | != | x != y | true, если x не равно y, в противном случае — false |

# 11 Логічні операції МП С++

|  |  |  |  |
| --- | --- | --- | --- |
| Оператор | Символ | Пример | Операция |
| Логическое НЕ | ! | !x | true, если x — false и false, если x — true |
| Логическое И | && | x && y | true, если x и y — true, в противном случае — false |
| Логическое ИЛИ | || | x || y | true, если x или y — true, в противном случае — false |

# 12 Операція присвоєння та її різновиди у МП С++

|  |  |  |  |
| --- | --- | --- | --- |
| Категория | Тип | Значение | Пример |
| Логический тип данных | bool | true или false | true |
| Символьный тип данных | char, wchar\_t, char16\_t, char32\_t | Один из ASCII-символов | ‘c’ |
| Тип данных с плавающей запятой | float, double, long double | Десятичная дробь | 3.14159 |
| Целочисленный тип данных | short, int, long, long long | Целое число | 64 |
| Пустота | void | Пустота |  |

# 13 Операції інкрементації та декрементації у МП С++

**Інкремент** — це операція збільшення, а **декремент** – зменшення значення змінної. Найчастіше ці операції змінюють значення на 1 (якщо змінна символьного типу, то береться наступний або попередній символ)

# 14 Реалізація послідовних алгоритмівмовою програмування С++. Способи введення даних в програму.

# 15 Введення даних в програму С++ у режимі діалогу. Способи виведення проміжних та кінцевих результатів обчислення.

# 16 Стандартні математичні функції та особливості їх використання для різних типів даних.

| **Функция** | **Описание** | **Пример** |
| --- | --- | --- |
| **abs( a )** | модуль или абсолютное значение от **а** | abs(-3.0)= 3.0 abs(5.0)= 5.0 |
| **sqrt(a)** | корень квадратный из **а,**причём **а**не отрицательно | sqrt(9.0)=3.0 |
| **pow(a, b)** | возведение **а**в степень **b** | pow(2,3)=8 |
| **ceil( a )** | округление **а** до наименьшего целого, но не меньше чем **а** | ceil(2.3)=3.0 ceil(-2.3)=-2.0 |
| **floor(a)** | округление **а** до наибольшего целого, но не больше чем **а** | floor(12.4)=12 floor(-2.9)=-3 |
| **fmod(a, b)** | вычисление остатка от  a/b | fmod(4.4, 7.5) = 4.4 fmod( 7.5, 4.4) = 3.1 |
| **exp(a)** | вычисление экспоненты **еа** | exp(0)=1 |
| **sin(a)** | **a** задаётся в радианах |  |
| **cos(a)** | **a** задаётся в радианах |  |
| **log(a)** | натуральный логарифм **a**(основанием является экспонента) | log(1.0)=0.0 |
| **log10(a)** | десятичный логарифм **а** | Log10(10)=1 |
| **asin(a)** | арксинус **a**, где **-1.0 < а < 1.0** | asin(1)=1.5708 |

# 17 Особливості запису арифметичних виразів у МП С++

# 18 Створення та реалізація програм за алгоритмами розгалуження мовою програмування С++. Оператор умовного переходу if ….. else.

**Cкладний оператор** — це послідовність операторів, відділених один від одного крапкою з комою, що починається з відкритої фігурної дужки { і закінчується закритою фігурною дужкою }:  
  
{ *оператор*1; *оператор*2; … *операторn*; };  
  
Компілятор сприйме складений оператор як один оператор.  
  
**Умовний оператор If** призначено для того, щоб задати вибір тієї чи іншої дії залежно від справдження певної умови. Цей оператор має такий синтаксис:  
  
if (*умова*) *оператор*1; else *оператор*2;

# 19 Реалізація повного і неповного розгалуження МП С++

# 20 Реалізація вкладеного розгалуження у МП С++

# 21 Тернарний оператор.

**yсловный (тернарный) оператор** (обозначается как ?:) является единственным тернарным оператором в языке С++, который работает с 3-мя операндами. Из-за этого его часто называют просто **«тернарный оператор»**.

|  |  |  |  |
| --- | --- | --- | --- |
| Оператор | Символ | Пример | Операция |
| sizeof | sizeof | sizeof(type) sizeof(variable) | Возвращает размер типа данных или переменной в байтах |
| Запятая | , | x, y | Вычисляется x, затем вычисляется y, а затем возвращается значение y |

# 22 Створення та реалізація програм за циклічними алгоритмами МП С++. Оператор циклу while … do

Цикл while є найпростішим з 4-х циклів, які є в мові C ++. Він дуже схожий на розгалуження if / else:

while (умова)

тіло циклу;

Цикл while оголошується з використанням ключового слова while. На початку циклу обробляється умова. Якщо його значенням є true (будь-нульове значення), то тоді виконується тіло циклу.

do  
    тело цикла;  
while (условие);

# 23 Створення та реалізація програм за циклічними алгоритмами МП С++. Оператор циклу do …while

# 24 Створення та реалізація програм за циклічними алгоритмами МП С++. Оператор циклу for. Гнучкість оператора for

# 25 Одновимірні масиви даних в МП С++. Формування одновимірного масиву МП С++. Ініціалізація одновимірного масиву

# 26 Виведення елементів одновимірного масиву МП С++

# 27 Реалізація алгоритму знаходження суми елементів одновимірного масиву в МП С++

# 28 Реалізація алгоритму знаходження добутку елементів одновимірного масиву в МП С++

# 29 Реалізація алгоритму знаходження кількості елементів одновимірного масиву в МП С++

# 30 Реалізація алгоритму знаходження максимального елемента одновимірного масиву та його порядкового номера в МП С++

# 31 Реалізація алгоритму знаходження мінімального елемента одновимірного масиву та його порядкового номера в МП С++

# 32 Реалізація алгоритму впорядкування одновимірного масиву методом «Бульбашки» в МП С++

# 33 Реалізація алгоритму впорядкування одновимірного масиву методом вибору в МП С++

# 34 Методи пошуку елемента одновимірного масиву. Двійковий пошук.

# 35 Методи пошуку елемента одновимірного масиву. Метод простого перебору.

# 36 Двовимірні масиви даних. Формування двовимірного масиву МП С++.

# 37 Знаходження максимального елемента в двовимірному масиві

# 38 Знаходження мінімального елемента в двовимірному масиві

# 39 Пошук максимального та мінімального по рядках та колонках двовимірного масиву

# 40 Обчислення суми в рядках і колонках двовимірного масиву

# 41 Обчислення добутку в рядках і колонках двовимірного масиву

# 42 Обчислення кількості в рядках і колонках двовимірного масиву. Впорядкування в рядках і колонках двовимірного масиву

# 43 Пошук в рядках і колонках двовимірного масиву.

# 44 Функції у мові програмування С++. Оголошення функції. Визначення функції.

# 45 Функції у мові програмування С++. Параметри формальні. Параметри фактичні.

# 46 Функції у мові програмування С++. Оператор return. Виклик функції.

# 47 Функції у мові програмування С++. Створення функцій користувача.

# 48 Робота з файлами у МП С++. Основи файлової системи.

# 49 Робота з файлами у МП С++. Вказівник файлу. Відкриття файлу. Закриття файлу.

# 50 Робота з файлами у МП С++. Запис символа. Зчитування символів з файлу.

# 51 Процедури роботи з файлами у МП С++. Робота з рядками. Видалення файлів.

# 52 Робота з файлами у МП С++. Очищення потоку. Стандартні потоки.

# 53 Робота з файлами у МП С++. Зв'язок з консольним вводом-виводом. Перенаправлення стандартних потоків.

# 54 Поняття про вказівники у МП С++. Оператори для роботи з вказівниками.

# 55 Вирази, що містять у собі вказівники: присвоєння вказівників, адресна арифметика, порівняння вказівників

# 56 Вказівники і масиви. Звичайна і непряма адресація. Ініціалізація вказівників.