Міністерство освіти й науки України
Запорізький національний університет

К.С. Решевська, А.О. Лісняк, С.Ю. Борю

ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

Навчальний посібник
для здобувачів ступеня вищої освіти бакалавра
спеціальності «Комп’ютерні науки» освітньо-професійної
програми «Комп’ютерні науки»

Затверджено
вченою радою ЗНУ
Протокол № 12
від 23.06.20

Запоріжжя
2020

УДК 004.32:004.007(075.8)
І-741

Решевська К.С., Лісняк А.О., Борю С.Ю. Об’єктно-орієнтоване програмування: навчальний посібник для здобувачів ступеня вищої освіти бакалавра спеціальності «Комп’ютерні науки» освітньо-професійної програми «Комп’ютерні науки». Запоріжжя : ЗНУ, 2020. с.94.

У навчальному посібнику представлено теоретичні матеріали з основ об’єктно-орієнтованого програмування, в яких розглянуто основні принципи даного підходу в програмуванні та їх реалізація на мові програмування С++.

Посібник складається з теоретичного матеріалу, контрольних запитань, практичних завдань та системи тестів для перевірки знань з основ об’єктно-орієнтованого програмування.

Навчальний посібник призначений насамперед для здобувачів вищої освіти спеціальності «Комп’ютерні науки», також він може бути корисним для технічних спеціалістів, які прагнуть отримати базові знання з об’єктно-орієнтованого прогамування та знати синтаксис програмного коду на мові С++.

Рецензент
С. В. Чопров, доктор технічних наук, доцент, професор кафедри програмної інженерії

Відповідальний за випуск
С.Ю. Борю, кандидат технічних наук, доцент, завідувач кафедри комп’ютерних наук

ВСТУП

Дисципліна «Об’єктно-орієнтоване програмування» належить до циклу професійної підготовки спеціальності та має статус «нормативна».

Метою викладання навчальної дисципліни «Об’єктно орієнтоване програмування» є оволодіння основними поняттями інформатики та комп’ютерної техніки, ознайомлення з сучасними поглядами на інформаційні процеси, засвоєння студентами головних принципів функціонування та використання ЕОМ як сучасного технічного засобу обробки інформації, оволодіння навичками алгоритмізації та програмування на мовах програмування для вирішення прикладних задач, вивчення мов програмування, вивчення парадигм об'єктно-орієнтованого та мета-програмування.

Основними завданнями вивчення дисципліни «Об’єктно-орієнтоване програмування» є:

– ознайомлення студентів з основними поняттями інформатики та комп’ютерної техніки, сучасними поглядами на інформаційні процеси, технічні та програмні методи їх супроводження;

– сприяння засвоєнню головних принципів функціонування та використання комп’ютеру, як засобу для автоматизації обробки інформації;

– отримання навичок та вмінь користання сучасними комп’ютерними засобами алгоритмізації та програмування сучасних прикладних задач на алгоритмічних мовах процедурного та об’єктному типу;

– вивчення студентами основних технологічних методів практичного застосування мовних засобів програмування для розробки програмного продукту, що призначений для практичного розв’язання задач інформаційного та математичного характеру.

– вивчення студентами мов програмування, сучасних поглядів на розробки програмного забезпечення та на інформаційні процеси, технічних та програмних методів їх супроводження;

– вивчення основних технологічних методів практичного застосування мовних засобів програмування для розробки програмного продукту.

У результаті вивчення курсу студенти повинні:

знати:

– предмет та головні поняття курсу;

– базові алгоритми і питання алгоритмізації;

– засоби запису алгоритмів;

– одержання програми, що виконується на ЕОМ, засобом трансляції;

– основні питання з організації технології програмування

– алгоритмічні конструкції та стандартні бібліотечні засобів мови програмування С++;

вміти:

– використовувати основні парадигми програмного забезпечення: структурну, об’єктно-орієнтовану, компонентну для розробки проекту комп’ютеризованої системи;

– володіти основами програмування та мовами різних рівнів: високого рівня, проблемно- та об’єктне-орієнтованими;

– розробляти програмне забезпечення комп’ютеризованої системи з використанням технологій програмування, заснованих на структурній та об’єктне орієнтованій парадигмі.

Дисципліна «Об’єктно-орієнтоване програмування» базується на знаннях та вміннях, отриманих під час вивчення дисципліни циклу професійної підготовки «Процедурне програмування».

Знання та уміння, отримані під час вивчення дисципліни «Об’єктно - орієнтоване програмування», можуть бути використані при вивчені таких дисциплін:

1. «Платформи корпоративних інформаційних систем»;
2. «Проектування програмних систем»;
3. «Програмування комп`ютерної графіки».

Навчальний посібник складається з наступних тем:

Тема 1. Вступ до технології програмування на мові C/C++.

Тема 2. Синтаксис і семантика операторів мови C/C++.

Тема 3. Об'єктно-орієнтоване програмування на мові C++.

Тема 4. Успадкування класів, віртуальне успадкування.

Тема 5. Віртуальні функції, абстрактне успадкування.

Тема 6. Перевантаження операцій.

Тема 7. Обробка виключних ситуацій.

Тема 8. Динамічне виділення пам'яті.

Тема 9. Бібліотека потокових класів C++.

Тема 10. Робота з текстовими файлами. Маніпулятори вводу-виводу.

Тема 11. Робота з бінарними файлами. POD-типи даних.

Тема 12. Шаблони функцій та класів. Бібліотеки шаблонів.

Тема 13. Стандартна бібліотека STL: контейнерні класи.

Тема 14. Стандартна бібліотека STL: ітератори, шаблонні алгоритми.

Кожна тема супроводжується переліком контрольних запитань і тестовими та практичними завданнями для перевірки знань і навичок з курсу.

РОЗДІЛ 1 ОБ'ЄКТНЕ ПРОГРАМУВАННЯ. ОСОБЛИВОСТІ ПРОЦЕДУРНОГО ТА ОБ'ЄКТНОГО ПРОГРАМУВАННЯ НА БАЗІ МОВИ С++

Тема 1 Вступ до технології програмування на мові С/С++

Ключові поняття: комп'терна програма, трансляція файлів, структура програми на мові С/С++, директива препроцесору, заголовні файли, коментарі, функції, формальні та фактичні параметри, оператори вводу/виводу в С++.

Існує багато визначень терміну «комп'терна програма», в стандартах ISO наведено наступні визначення цього поняття.

Комп'ютерна програма – комбінація комп'ютерних інструкцій і даних, що дозволяє апаратного забезпечення обчислювальної системи виконувати обчислення або функції управління (стандарт ISO / IEC / IEEE 24765: 2010).

Комп'ютерна програма – синтаксична одиниця, яка відповідає правилам певної мови програмування, що складається з визначень та операторів або інструкцій, необхідних для певної функції, завдання або вирішення проблеми (стандарт ISO / IEC 2382-1: 1993).

1.1 Структура програми на мові С++

Кожна програма на мові програмування високого рівня повинна бути оформлена відповідно до правил цієї мови. Перш ніж приступати до написання програм, необхідно вивчити структуру програм на тій мові програмування, на якій вона буде написана.

Структура програми – це розмітка робочої області (області коду) з метою чіткого визначення основних блоків програм і синтаксису.

Програма на мові програмування С++ може складатись з одного або декількох файлів. Файли транслюються С-компілятором незалежно один від одного і потім об'єднуються програмою-будівником завдань, в результаті чого створюється файл з програмою, яка є готовою до виконання.

Трансляція файлів – перетворення програми, яка написана на одній з мов високого рівня, в програму, що складається з машинних команд.

Файли, що містять код програми, називаються вихідними. У мові С++ вихідні файли бувають двох типів:

– заголовні (h-файли);

– файли реалізації (С-файли).

Імена заголовних файлів мають розширення ".h". Імена файлів реалізації мають розширення ".c".

Заголовні файли містять:

– опис оголошень функцій;

– імена і типи зовнішніх змінних,

– константи;

– нові типи;

– структури.

При трансляції заголовних файлів, як правило, ніякі об'єкти не створюються. Файли реалізації містять тексти функцій і визначення глобальних змінних.

! Подання вихідних текстів у вигляді заголовних файлів і файлів реалізації необхідно для створення великих проектів, що мають модульну структуру. Заголовки служать для передачі інформації між модулями.

Заголовні файли підключається за допомогою директиви препроцесора #include.

Препроцесор – це спеціальна програма, яка є частиною компілятора мови Сі. Вона призначена для попередньої обробки тексту програми.

Наприклад, стандартні функцій вводу-виводу підключаються наступним чином:

#include <iostream.h>

Ім'я h-файлу записується в кутових дужках, якщо цей h-файл є частиною стандартної С–бліотеки і розташований в одному з системних каталогів. Імена h-файлів, створених під час розробки проекту і розташованих в поточному каталозі, вказуються в подвійних лапках:

#include "mylib.h"

Мова С++ є блочно-структурованою. Кожен блок розміщується в фігурних дужках {}.

Основним блоком у програмі консольного додатку на мові С++ є головна функція, що має ім'я main ().Функцію main () можна викликати з інших функцій програми, вона є керуючою.

Наступні за ім'ям функції круглі дужки призначені для вказівки параметрів (аргументів), які передаються в функцію при зверненні до неї. В даному випадку операційна система не передає в функцію main () ніяких аргументів, тому список аргументів у круглих дужках порожній.

Вихідний код може містити коментарі. Коментарі дозволяють зрозуміти змив програми, що роблять ті чи інші її частини. При компіляції коментарі ігноруються і не мають жодного впливу на роботу програми та на його розмір.

У мові C++ є два типи коментарів: однорядковий і багаторядковий. Однорядковий коментар розміщується на одному рядку після подвійного слешу //:

```
#include <iostream>

int main()
{
   cout << "Hello World!";   // виводимо рядок на консоль
   return 0;                 // виходимо з функції
}                            // кінець функції
```

Складений коментар укладається між символами / * текст коментаря * /. Він може розміщуватися на декількох рядках. наприклад:

```
#include <iostream>
/*
   Визначення функції Main
   Виводить на консоль рядок Hello World!
*/
int main()
{
   cout << "Hello World!";
   return 0;
}
```

1.2 Функції в мові програмування C++

Програма на мові C++ складається з однієї або більше підпрограм, які називаються функціями.Функція є основною структурною одиницею мови C++.

✎ ***Функція*** – це підпрограма, яка може маніпулювати даними і повертати деяке значення. Кожна програма мовою C ++ має щонайменш одну функцію (main), яка виконується при запуску програми автоматично та може викликати інші функції; ті, в свою чергу, можуть викликати наступні.

Функція зазвичай виконує алгоритм, який описується і реалізується окремо від інших алгоритмів. При виконанні функції передаються аргументи, які можуть бути використані в функції. Виклик функції відбувається в

результаті використання її імені у виразі. За ім'ям функції слідують круглі дужки, всередині яких перераховуються фактичні значення її аргументів.

! Навіть якщо аргументів у функції немає, круглі дужки з порожнім списком аргументів обов'язково повинні бути присутніми.

Після виклику функції значення, повернене в результаті її виконання, використовується у виразі (ім'я функції замінюється повернутим значенням).

Оголошується функція, аналогічно оголошенню змінної, вказується ім'я функції, тип значення, яке вона може повертати, набір її параметрів (для кожного параметра задається тип і, при бажанні, ім'я).

Оголошення функції називають також її **прототипом**.

Тип_результату ім'я_функції (Тип_парарматру1, Тип_параметру2, ...);

Тип_результату – деякий існуючий (наприклад, вбудований) тип даних або ключове слово void, яке вказує на те що функція ніякого значення повертати не буде

Ім'я_функції – унікальний для даного простору імен ідентифікатор.

Тип_параметруN – деякий існуючий тип даних.

Визначення або опис функції містить перелік тих операцій, які будуть проводиться всередині функції:

Тип_результату ім'я_функції (Тип_параметру1 Ім'я_параметру1, Тип_параметру2 Ім'я_параметру2, ...)
{ оператор1;
оператор2; ...
ОператорN;
return n; }

Ім'я_параметруN – унікальне всередині функції ім'я N-го параметру.

ОператорN – деякі оператори і вирази, що містяться всередині функції і виконуються кожного разу при виконанні функції.

return n – оператор, який зупиняє роботу функції і повертає n в якості її значення (при цьому тип n повинен відповідати типу результату в оголошенні функції). Наявність цього оператору обов'язково для функції, яка повертає значення. Для функції оголошеної як void можна викликати оператор return без аргументів, це достроково завершить функцію, інакше – будуть виконані всі дії до кінця блоку опису функції.

Тіло функції – блок визначення функції.

! Одна функція не може оголошуватися або визначатися всередині іншої (тобто не можна оголошувати і визначати функції всередині main).

Приклад оголошення і опису функції:

```
int max (int, int);
int max (int n1,int n2) {
  if(nl > n2) {
    return n1;
  } else {
    return n2;
  }}
int main(void) {
  int a = 100 - max(10,20);
  cout << a;
  return 0;
}
```

Видно, що вся інформація, що була в прототипі функції, повторюється в її визначенні, тому якщо функція визначена до її першого виклику, то окремо прототип вказувати не обов'язково.

Приклад:

```
double cube (double a) {
  return a*a*a;
}

int main(void) {
  double pi = 3.1415;
  cout << cube(pi);
  return 0;
}
```

Але окремо прототип вказують в тих випадках, коли функція буде описуватися пізніше свого використання. Наприклад, можна було оголосити функцію до main, викликати її з main, але описати тільки після main.

Приклад:

```
double cube (double);
int main(void) {
  double pi = 3.1415;
  cout << cube(pi);
  return 0;
}

double cube (double a) {
  return a*a*a;
}
```

Параметри функції розділяються на формальні та фактичні.

Формальні параметри – параметри, які існують в прототипі і тілі визначення функції. Вони задаються деякими унікальними іменами і всередині функції доступні як локальні змінні.

Фактичні параметри – параметри, які існують в основній програмі. Вони вказуються при виклику функції на місці формальних.

У момент виклику функції значення фактичних параметрів присвоюються формальним. Відповідно, імена формальних і фактичних параметрів можуть збігатися, це не викличе конфлікту.

Параметри функцій передаються та повертаються декількома способами:

– за **значенням** – передача значень формальних параметрів у якості фактичних;

– за **посиланням** – передача адреси формальних параметрів у якості фактичних;

Приклад:

```
#include <stdlib.h>
#include <stdio.h>
//Оголошення прототипи функцій користувача

//передача параметрів за значенням
int Sum(int, int);
//передача параметрів за значенням та за посиланням
void Mul(int, int, int&);

//основна програма
int main(int argc, char* argv[])
{
  int  x = 3,
       y = 5,
       z = 0;
  z = Sum(x,y);
  printf("Sum %d and %d = %d\n",x,y,z);
  Mul(x,y,z);
  printf("Mul %d and %d = %d\n",x,y,z);
  system("pause");
  return 0;
}

//Визначення функцій
int Sum(int a, int b){
  return a+b;
}
void Mul(int a, int b, int& res){
  res = a*b;
```

```
}
```

Результат роботи програми:

```
Sum 3 and 5 = 8
Mul 3 and 5 = 15
```

1.3 Оператори вводу/виводу в C++

В C ++, як і в C, немає вбудованих в мову засобів вводу/виводу.
В C для цих цілей використовується бібліотека **stdio.h**.
В C ++ розроблена нова бібліотека вводу/виводу **iostream**, що використовує концепцію об'єктно-орієнтованого програмування:

```
#include <iostream>
```

Бібліотека **iostream** визначає три стандартні потоки:
– cin стандартний вхідний потік (stdin в C);
– cout стандартний вихідний потік (stdout в C);
– cerr стандартний потік виведення повідомлень про помилки (stderr в C).

!Для їх використання в Microsoft Visual Studio необхідно прописати рядок:

```
using namespace std;
```

Для виконання операцій вводу/виводу перевизначені дві операції порозрядного зсуву:
>> отримати з вхідного потоку;
<< помістити у вихідний потік.

Приклад:

```
#include <iostream>

int main()
{
   int age;
   double weight;
   cout << "Input age: ";
   cin >> age;
   cout << "Input weight: ";
   cin >> weight;
   cout << "Your age: " << age << " your weight: " << weight;
   return 0;
```

```
}
```

Результат роботи програми:

```
Input age: 23
Input weight: 45
 Your age: 23Your weight: 45
```

Рядки можуть містити керуючі послідовності, які інтерпретуються певним чином (таблиця 1.1). Також оператори << можна завершувати значенням **endl,** яке викликає перехід на новий рядок і звільнення буферу. При виведенні в потік дані спочатку поміщаються в буфер. І звільнення буферу гарантує, що всі передані для виведення на консоль дані негайно будуть виведені на консоль.

Таблиця 1.1– Керуючі послідовності

\r	повернення коретки в початок рядка
\n	перехід на новий рядок
\t	горизонтальна табуляція
\v	вертикальна табуляція
\a	сигнал биперу (спікеру комп’ютеру)

Використаємо керуючі послідовності у попередьому прикладі.

Приклад :

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

int main()
{
   int age;
   double weight;
   cout << "Input age: ";
   cin >> age;
   cout << "Input weight: ";
   cin >> weight;
   cout << "\v Your age: " << age <<"\n"<< "\t Your weight: " << weight;
   return 0;
}
```

Результат роботи програми:

```
Input age: 23
Input weight: 45
        Your age: 23
        Your weight: 45
```

❓Контрольні запитання:

1. Як ви розумієте поняття «комп'ютерна програма»?
2. Які визначення поняттю «комп'ютерна програма» наведено у стандартах ISO?
3. Що розуміється під поняттям «трансляція файлів»?
4. Яку структуру має програма, написана на мові програмування С++?
5. З яких файлів складається проект, написаний на С++?
6. Який синтаксис функції на С++?
7. Що таке формальні та фактичні параметри функції?
8. За допомогою яких операторів здійснюються вводу/виводу у С++?
9. Чи можливо створити підпрограму яка повертає декілька значень?
10. Що таке прототип функції?
11. Які варіанти визначення прототипів існують та який з них найкращій?
12. Як визначити підпрограму яка нічого не повертає?

✍Практичні завдання

1. Написати програмний код на мові програмування С++ з використанням функцій для вирішення наступної задачі: для цілочисельної прямокутної матриці визначити кількість рядків, що не містять жодного нульового елементу та максимальне з чисел, що зустрічаються в заданій матриці більше одного разу.

2. Написати програмний код на мові програмування С++ з використанням функцій для вирішення наступної задачі: для заданої цілочисельної квадратної матриці знайти максимум серед сум елементів діагоналей, що паралельні головній діагоналі.

3. Написати програмний код на мові програмування С++ з використанням функцій для вирішення наступної задачі: починаючи з елементу a00 обійти по спіралі квадратну матрицю NxN, роздрукувавши елементи в порядку обходу

4. Написати програмний код на мові програмування С++ з використанням функцій для вирішення наступної задачі: знайти максимальний серед всіх елементів тих рядків заданої матриці, які впорядковані (або за збільшенням, або за зменшенням).

5. Написати програмний код на мові програмування С++ з використанням функцій для вирішення наступної задачі: по заданій квадратній матриці NxN побудувати вектор завдовжки 2(N-1)x1, елементи якого – максимуми елементів діагоналей, паралельних головній.

6. Написати програмний код на мові програмування С++ з використанням функцій для вирішення наступної задачі: ущільнити задану матрицю,

видаляючи з неї рядки і стовпці, заповнені нулями, знайти номер першого з рядків, що містить хоч би один позитивний елемент.

7. Написати програмний код на мові програмування C++ з використанням функцій для вирішення наступної задачі: для цілочисельної прямокутної матриці визначити кількість рядків, що містять хоча б один нульовий елемент; номер стовпця, в якому знаходиться найдовша серія однакових елементів.

8. Написати програмний код на мові програмування C++ з використанням функцій для вирішення наступної задачі: для заданої матриці розміром NxN знайти таке k, що k-й рядок матриці співпадає з k-м стовпцем.

9. Написати програмний код на мові програмування C++ з використанням функцій для вирішення наступної задачі: починаючи з центру, обійти по спіралі всі елементи матриці NxN, роздрукувавши їх в порядку обходу.

10. Написати програмний код на мові програмування C++ з використанням функцій для вирішення наступної задачі: для заданої матриці розміром NxN знайти таке k, що k-й рядок матриці співпадає з k-м стовпцем.

Тести з теми «Вступ до технології програмування на мові C/C++»

1. Що означає передача параметрів функції за значенням?
   А.
   зміна формальних параметрів функції на змінює значення фактичних;
   Б.
   зміна формальних параметрів функції на значення фактичних;

2. Оберіть вірне визначення функції:
   А.
   ```
   void function(int)
   {
   cout << "Hello"
   }
   ```
   Б.
   ```
   void function(x)
   {
   cout << "Hello"
   }
   ```
   В.
   ```
   int funct(int x)
   {
   return x = x + 1;
   ```

}

3. Вкажіть тип значення, яке повертає функція *int func(char x, float v, double t);*
   А. *int*
   Б. *float*
   В. *char*
   Г. *double*

4. Вкажіть вірний виклик функції, яка попередньо визначена
   А.
   int funct();;
   Б.
   funct();
   В.
   funct;
   Г.
   funct x, y;

5. Що з переліченого не є прототипом функції?
   А.
   int funct(char x, char y);
   Б.
   char x();
   В.
   void funct();
   Г.
   double funct(char x)

Тема 2 Синтаксис і семантика операторів мови С/С++

Ключові поняття: оператор, складені оператори, оператори вибору, оператори циклів, оператори переходу.

Інструкція або оператор (англ. Statement) [1] – найменша автономна частина мови програмування; команда або набір команд. Програма звичайно являє собою послідовність інструкцій.

Різновиди операторів:
– складені оператори;
– оператори вибору;
– циклічні оператори;
– оператори переходу.

1.1. Складені оператори

Складені оператори – оператори, які складаються з блоків операторів.

Блок операторів – це послідовність операторів, укладена в фігурні дужки.

Приклад:

```
if(i > 0)
{
mas[i] = x;          складений оператор
x++;
}
```

1.2. Оператори вибору

Оператори вибору – це умовний оператор і перемикач.

Умовний оператор – оператор **if**, він має повну і скорочену форму:
– повна форма: *if (<вираз-умова>) <оператор1>; else <оператор2>;*
– скорочена форма: *if (<вираз-умова>) <оператор>;*

Приклад повної форми оператору **if**:

```
if (x<y)
  max=y;
else
  max=x;
```

Приклад скороченої форми оператору **if**:

```
if (n<m)
  set=n;
```

Перемикач – оператор, який здійснює множинний вибір:

switch (<выражение>)
{
case <константа1> : <оператор1 >;
case <константа2> : <оператор2 >;
. . . . . .
default: <оператори>;

При виконанні оператору switch, обчислюється вираз, який записано після switch і його значення послідовно порівнюється з константами після case. При першому ж збігу виконуються оператори помічені даною міткою. Якщо серед перелічених констант немає обчисленого значення, то виконуються оператори, які слідують за міткою **default**. Мітка default може бути відсутня.

Приклад:
switch (n)
{
case 1 : cout<< "число=1";
case 2 : cout<< " число=2";
case 3 : cout<< " число=3";
case 4 : cout<< " число=4";
default: cout<< "Кінець роботи програми";
}

1.3. Циклічні оператори

Циклічні оператори – оператори, в яких виконуються один або декілька операторів циклічно до тих пір, поки лічильник відповідає певній заданій умові.

Цикл з передумовою – циклічний оператор, в якому перевірка відповідності лічильника заданій умові здійснюється до входження в цикл:

while (<вираз-умова>)
<Тіло_цикла>;

Приклад:
while (i < 1000)
{ i++; sum += i; }

Цикл з постумовою – циклічний оператор, в якому перевірка відповідності лічильника заданій умові здійснюється після виходження з циклу:

do
<Тіло_цикла>;
while (<вираз-умова>);
Тіло циклу виконується до тих пір, поки <вираз-умова> істинно.

Приклад:
do{ i++; sum += i; }
while (i < 10);

Цикл з параметром – циклічний оператор з параметром:
for (<вираз_1>; <вираз-умова>; <вираз_2>)
тіло_цикла;

Приклад:
for (i = 1; i <= 10; i++)
{. sum = sum + i; }

1.4. Оператори переходу

Оператор переходу – оператор, який виконує виконує безумовну передачу управління:

– **break** – оператор переривання циклу:
{
<Оператори>
if (<вираженіе_условіе>) break;
<Оператори>
}

Оператор break доцільно використовувати, коли умову продовження ітерацій треба перевіряти в середині циклу.

Приклад:
for (s = 0, i = 1; i <100; i ++)
{
cin >> x;
if (x == 0) break;
s + = x;
}

– **continue -** перехід до наступної ітерації циклу. Він використовується, коли тіло циклу містить розгалуження.

Приклад:
for (k = 0, s = 0, x = 1; x! = 0;)
{
cin >> x;
if (x <= 0) continue;
k ++; s + = x;
}

? Контрольні запитання:

1. Як ви розумієте поняття «оператор» та «складений оператор»?
2. Чим відрізняється повна та скорочена форми умовного оператору?

3. В якому випадку використовується циклічний оператор з передумовою, а коли – з постумовою?

4. Для чого використовуються оператори переходу?

✍Практичні завдання:

1. Використовуючи один з операторів циклу, знайти суму цілих додатних чисел, кратних 3 і менших 200.

2. Використовуючи один з операторів циклу, знайти суму цілих додатних парних чисел, менших за 100.

3. Використовуючи один з операторів циклу, знайти суму цілих додатних непарних чисел, менших за 200.

4. Використовуючи один з операторів циклу, знайти суму цілих додатних чисел, більших за 20 та менших за 100 і кратних 3

5. Використовуючи один з операторів циклу, знайти суму ряду з точністю ε = 10-4, загальний член якого: $a_n = \frac{(-1)^{n-1}}{n^n}$

6. Використовуючи один з операторів циклу, знайти суму ряду з точністю ε = 10-4, загальний член якого: $a_n = \frac{1}{2^n} + \frac{1}{3^n}$

7. Використовуючи один з операторів циклу, знайти суму ряду з точністю ε = 10-4, загальний член якого: $a_n = \frac{1}{((3n-2)(3n+1))}$

8. Використовуючи один з операторів циклу, знайти суму ряду з точністю ε = 10-4, загальний член якого: $a_n = \frac{(2n-1)}{2^n}$

9. Використовуючи один з операторів циклу, знайти суму ряду з точністю ε = 10-4, загальний член якого: $a_n = \frac{10^n}{n!}$

10. Використовуючи один з операторів циклу, знайти суму ряду з точністю ε = 10-4, загальний член якого: $a_n = \frac{n!}{(2n)!}$

Тести до теми «Синтаксис і семантика операторів мови С/С++»

1. Оберіть вірне визначення поняттю «оператор».

А.*найменша автономна частина мови програмування;*

Б.*команда або набір команд;*

В. *набір команд, які які об'єднані у {}.*

2. Який з наведених прикладів використання умовного оператору містить помилку?

А. *if (a==b) m=a else max=x;*
Б. *if (a<b) m=a; else max=x;*
В. *if (a>b) m=a; else max=x;*

3. Якого значення набуде змінна b у результаті виконання оператору switch:

```
switch ( 4! )
{
case 4 : b=5;
case 24 :  b=4;
case 16 :  b=6;
case 20 :  b=7;
default: cout<< “Кінець роботи програми”;
}?
```

А. *4;*
Б. *5;*
В. *6;*
Г. *7.*

4. В якій з реалізації оператору for немає помилок?

А.
```
While X>0 do;
X:=X-5;
```
Б.
```
While do;
X:=X-5;
```
В.
```
While X>0 do
X:=X-5;
```

5. Скільки разів буде виконано обчислення у тілі циклу m:=5; While m<=10 do m:=m+2?

А. *три;*
Б. *жодного разу;*
В. *два;*
Г. *чотири.*

6. Яке значення набуде змінна z після виконання оператору for:

```
Z:=0;
For i:=1 to 3 do
Z:=Z+2*i;
```

А. *12;*
Б. *10*;
В. *11;*

Г. *14.*

7. Який рядок необхідно додати до коду, щоб не відбулось зациклення:
```
S:=0; i:=1;
While S<10 do
begin
i:=i+i;
end;
```
А. *S:=S+1*;
Б. *S:=S/S+1*;
В. *S:=S-1;*
Г. *S:=S^2*.

8. Які з перелічених операторів є операторами циклу?
А. *While*;
Б. *Do while*;
В. *Case;*
Г. *Switch*.

9. Скільки разыв виконається тіло циклу у наступному коді?
```
int i = 10;
 while (i < 50)
{
   if (i < 0)
   {
      break;
   }
   i = i - 5;
}
```
А. *три*;
Б. *два*;
В. *жодного разу;*
Г. *нескінченну кількість разів*.

10. Скільки разів виконається тіло циклу у наступному коді?
```
int i = 0;
 while (i < 5)
{
   if (i == 3)
   {
      i = i + 1; continue;
   }
      i = i + 1;}
```
А. *чотири;*

Б. *п’ять*;
В. *жодного разу;*
Г. *нескінченну кількість разів.*

Тема 3 . Об'єктно-орієнтоване програмування на мові С++

Ключові поняття: об’єктно-орієнтоване програмування, об’єкт, приципи об’єктно-орієнтованого програмування: інкапсуляція, поліморфізм, успадкування, властивості та методи класу, конструктор та деструктор, модифікатори доступу.

1.1 Вступ до об’єктно-орієнтованого програмування

Об’єктно-орієнтоване програмування – методологія програмування, яка заснована на представленні програми у вигляді сукупності об’єктів, кожен з яких є реалізацією певного класу, а класи утворюють ієрархію за принципами успадкування.

Кожен об’єкт при програмуванні задається класом.

Клас – фундаментальне поняття, яке визначає користувацький тип даних, який об’єднує ці дані та методи їх обробки.

Окремий представник класу є його екземпляром. Наприклад:

- стіл – це класс;
- дерев’яний стіл білого кольору – екземпляр класу, т.т. окримий його представник.

Класи в програмуванні складаються з властивостей і методів.

Властивості класу – це будь-які дані, якими можна характеризувати об'єкт класу.

У вище наведеному прикладі властивостями об’єкту «стіл» буде колір, розмір, матеріал, з якого виготовлено стіл.

Для виконання поставленої задачі над властивостіми об’єкту необхідно виконувати певні дії, вони називаються **методами класу**.

Методи – це функції, які можуть виконувати будь-які дії над даними (властивостями) класу.

Наприклад, для столу необхідно розрахувати кількість деревини, необхідної для його побудови. Властивостями класу будуть дані, що характеризують стіл, а методами – функції, які розраховують кількість деревини для столу певного розміру.

! Об’єктно-орієнтоване програмування бається на трьох принципах: інкапсуляція, поліморфізм, успадкування.

Інкапсуляція – здатність об'єкту приховувати внутрішній устрій своїх властивостей і методів.

Згідно з цим принципом, клас повинен розглядатися як чорний ящик. Зовнішній користувач не знає деталі реалізації об'єкта і працює з ним тільки шляхом наданого об'єктом інтерфейсу.

Успадкування – принцип, який дозволяє описати новий клас на основі вже існуючого батьківського (базового) класу.

Клас-нащадок може додати свої власні властивості і методи, користуватися методами і властивостями базового класу. Спадкування дозволяє будувати ієрархії класів.

Поліморфізм – явище, при якому класи-нащадки можуть змінювати реалізацію методу класу-предку, зберігаючи його інтерфейс.

Поліморфізм дозволяє обробляти об'єкти класів-нащадків як однотипні об'єкти, не дивлячись на те, що реалізація методів у них може відрізнятися.

1.2 Модифікатори доступу public та private

Всі властивості і методи класів мають права доступу. За замовчуванням, весь вміст класу є доступним для читання і запису тільки для нього самого. Для того, щоб дозволити доступ до даних класу ззовні, використовують модифікатор доступу **public**. Всі функції і змінні, які знаходяться після модифікатора **public**, стають доступними з усіх частин програми.

Закриті дані класу розміщуються після модифікатору доступу **private**.

Захищені члени класу – можуть використовуватися тільки членами класу та його нащадками та вказуються після модифікатору **protected**.

! Зазвичай, приватними роблять всі властивості класу, а публічними – його методи. Всі дії з закритими властивостями класу реалізуються через його методи.

Приклад:

```
class NewYear
{
private:
        int current_year;
        int the_next_year;
public:
void set(int x, int y)
  {
  current_year=x;
 the_next_year=y;
 }
        void show()
        {
                cout << "Поточний рік:" << current_year << endl;
                cout << "Наступний рік:" << the_next_year << endl;
        }};
```

Для використання заданого класу, т.т. для використання заданого користувацького типу даних та методів, які можна виконувати над екземплярами класу, у головній програмі оголошуються змінні та визиваються функції класу, як показано у наступному прикладі:

```
int main()
{       NewYear n1;
```

```
n1.set(2019, 2020);
n1.show();
}
```

Змінна *current_year* набуде значення 2019, а змінна *the_next_year* – 2020. Обидва значення виведедуться на консоль.

1.3 Конструктор та деструктор

Коли створюються елементи (змінні), присвоїти їм значення в самому визначенні класу немає можливості, бо компілятор видасть помилку. Тому необхідно створювати окремий метод (так звану set-функцію) класу, за допомогою якого і буде відбуватися ініціалізація елементів. При цьому, якщо необхідно створити, наприклад, 20 об'єктів класу, то щоб ініціалізувати елементи потрібно 20 разів викликати set-функції. В таких випадках використовується конструктор.

Конструктор – це особливий тип методу класу, який автоматично викликається при створенні об'єкта цього ж класу.

Конструктор копіювання – конструктор, що приймає в якості аргументу об'єкт того ж класу, або посилання на нього та має наступну сигнатуру: *назва_класу (const назва_класу &);*

Якщо у класі не визначено жодного конструктора компілятор автоматично створює конструктор за замовчення та копіювання. Але у разі явного визначення будь якого конструктора конструктор за замовченням компілятор не надає.

Для звільнення пам’яті, зайнятої значеннями при ініціалізації змінних у конструкторі, використовують десктрутор.

Деструктор (від слова destruct – руйнувати) – спеціальний метод класу, який служить для знищення елементів класу. Найчастіше його використовують тоді, коли в конструкторі, при створенні об'єкта класу, динамічно була виділена ділянка пам'яті і необхідно цю пам'ять очистити, якщо ці значення вже не потрібні для подальшої роботи програми.

! Важливо запам'ятати:

– конструктор і деструктор завжди оголошуються в розділі *public*;

– при оголошенні конструктору, тип даних, що повертається значення не вказується, в тому числі – void;

– у деструктора також немає типу даних для значення, що повертається, до деструктора не можна передавати ніяких параметрів;

– ім'я деструктора ідентично імені конструктора, але з приставкою ~;

– у класі допустимо створювати кілька конструкторів, якщо це необхідно. Імена будуть однаковими. Компілятор буде їх розрізняти за переданим параметрам.

– в класі може бути оголошений лише один деструктор.

Приклад:

```
#include <iostream>
using namespace std;
class NewYear
{
private:
	int current_year;
	int the_next_year;
public:
// Конструктор без параметрів
		NewYear():current_year(0), the_next_year(0)
	{
			unix_year = 1969;
	}

// Конструктор з двома параметрами
		NewYear(int c, int n) : current_year(c), the_next_year(n)
		{ }
// Деструктор
		~NewYear()
		{
		cout << "Спрацював деструктор" << endl;
		}
	void show()
	{
		cout << "Поточний рік:" << current_year << endl;
		cout << "Наступний рік:" << the_next_year << endl;
		}
	};
int main()
{
	setlocale(0, "");
	NewYear n1;// викликається конструктор без параметрів
	NewYear n2(2019,2020); //викликається конструктор з параметрами
	cout << "Змінна  n1:" << endl;
	n1.show();
	cout << "Змінна n2:" << endl;
	n2.show();
	return 0;
}
```

Результат роботи програми:

```
Зм?нна n1:
Поточний р?к:0
Наступний р?к:0
Зм?нна n2:
Поточний р?к:2019
Наступний р?к:2020
Спрацював деструктор
Спрацював деструктор
```

При оголошенні змінної *n1* спрацював конструктор за замовчуванням, при оголошенні змінної *n2* спрацював конструктор з параметрами. Деструктор спрацював момент, коли завершилася робота програми.Він спрацював два рази: перший деструтор видалив другий створений об'єкт **(***n2*) , а потім перший (*n1*).

? Контрольні запитання:

1. Як ви розумієте поняття «об’єктно-орієнтоване програмування »?
2. Наведіть приклади об’єктів та їх властивості і визначте декілька екземплярів. Які практичні задачі вирішуються на таких об’єктах?
3. Які модифікатори в С++ задаються приватний та публічний доступ до властивостей та методів?
4. Як називають дані та функції визначені у класі?
5. З якою метою створюють конструктор та деструктор классів?

✍Практичні завдання:

Напишіть програмний код на мові програмування С++, у якому створюються, використовуються, знищуються об'єкти, визначеного користувачем типу (класу). Реалізуйте визначення нового класу, в якому обов'язково повинні бути присутніми :

- конструктори та деструктори;
- один або декілька методів ініціалізації – *Init()*, що повинні контролювати значення аргументів на коректність;
- метод вводу з клавіатури – *Read()*;
- метод виводу на екран *Display()*.

Опис властивостей та методів об’єктів:

1. Поле first – дробове число; поле second - ціле число, показник ступеня. Реалізувати метод power() – піднесення числа first до степеня second. Метод повинен правильно працювати при будь-яких припустимих значеннях first і second.

2. Поле first – дробове число; поле second – дробове число, показник степеня. Реалізувати метод power() – піднесення числа first до степеня second. Метод повинен правильно працювати при будь-яких припустимих значеннях first і second.

3. Поле first – ціле додатне число, чисельник; поле second – ціле додатне число, знаменник. Реалізувати метод part() – виділення цілої частини дробу first/second. Метод повинен перевіряти нерівність знаменника нулю.

4. Поле first – ціле додатне число, номінал купюри; номінал може приймати значення 1, 2, 5, 10, 50, 100, 500, 1000, 5000. Поле second – ціле додатнє число, кількість купюр даного номіналу. Реалізувати метод summa – обчислення грошової суми.

5. Поле first – дробове додатне число, ціна товару; поле second – ціле додатне число, кількість одиниць товару. Реалізувати метод cost() – обчислення вартості товару.

6. Поле first – ціле додатне число, калорійність 100 г продукту; поле second - дробове додатне число, маса продукту в кілограмах. Реалізувати метод power() - обчислення загальної калорійності продукту.

7. Поле first – дробове число, ліва границя діапазону; поле second – дробове число, права границя діапазону. Реалізувати метод rangecheck() – перевірку заданого числа на приналежність діапазону.

8. Поле first – ціле число, ліва границя діапазону включена в діапазон; поле second – ціле число, права границя діапазону не включена в діапазон. Пара чисел представляє напіввідкритий інтервал [first, second). Реалізувати метод rangecheck() – перевірку заданого цілого числа на приналежність діапазону.

9. Поле first – ціле додатне число, години поле second - ціле додатне число, хвилини. Реалізувати метод minutes – приведення часу в хвилини.

10. Лінійне рівняння y = Ax + B. Поле first — дробове число, коефіцієнт A; поле second – дробове число, коефіцієнт B. Реалізувати метод function() — обчислення для заданого x значення функції y.

11. Лінійне рівняння y = Ax + B. Поле first – дробове число, коефіцієнт A; поле second – дробове число, коефіцієнт B. Реалізувати метод root() – обчислення кореня лінійного рівняння. Метод повинен перевіряти нерівність коефіцієнта B нулю.

12. Поле first – дробове число, координата x точки на площині; поле second – дробове число, координата y точки на площині. Реалізувати метод distance() – відстань від точки то початку координат.

13. Поле first – дробове додатне число, катет a прямокутного трикутника; поле second – дробове додатне число, катет b прямокутного трикутника. Реалізувати метод hypotenuse() – обчислення гіпотенузи.

14. Поле first – дробове додатне число, оклад; поле second – ціле число, кількість відпрацьованих днів у місяці. Реалізувати метод summa() – обчислення нарахованої суми за дану кількість днів для заданого місяця: оклад / дні_місяця * відпрацьовані_дні.

15. Поле first ціле додатне число, тривалість телефонної розмови в хвилинах; поле second – дробове додатне число, вартість однієї хвилини в рублях. Реалізувати метод cost() – обчислення загальної вартості розмови.

Тести з теми «Об'єктно-орієнтоване програмування на мові С++»

1. Оберіть вірне визначення поняттю «об'єкт»?

А. *об'єкт – це інструмент боротьби зі складністю різних систем реальних сутностей з характеристиками – агрегація, залежність, конкретизація*;

Б. *об'єкт – це конкретне уявлення абстракції з характеристиками - модифікатор, селектор, літератор*;

В. *об'єкт – це конкретне уявлення абстракції, яке володіє індивідуальністю, станом і поведінкою*.

2. Яке оголошення екземпляру класу *tStudent* викличе помилку?

```
Class tStudent
{
private:
int age;
public:
tStudent(int _age)
{
age=_age;
}
~tStudent{}
};
```

А. *tStudent s1;*

В. *tStudent s2(17)*;

Г. *s2.age=17*;

Д. *tStudent s2(16.4);*

Е. *tStudent age(15).*

3. Принцип об'єктно-орієнтованого програмування, який полягає в об'єднанні атрибутів і методів об'єкту з метою забезпечення збереження даних, називається:

А. *спадкування*;

Б. *поєднання;*

В. *ініціалізація;*

Г. *інкапсуляція.*

4. Яка функція виконує початкову ініціалізацію даних в класі?

А. *немає вірної відповіді*;

Б. *конструктор*;

В. *деструктор*.

5. Оберіть вірне твердження стосовно деструктора класу в С ++.

А. *деструктор приймає у якості параметру адресу того об'єкту, який потрібно знищити*;

Б. *деструктор приймає як параметр покажчик this*;
В. *деструктор не містить параметрів*.

6. Що потрібно додати в клас А, щоб програма скомпілювалась успішно?

```
#include <iostream>
class A
{
public:
 A (const int num): _ num (num) {}
private:
int _num;};
int main (void)
{
A a;retur n 0;}
```

А. *конструктор копіювання*;

Б. *конструктор за замовчуванням*;

В. *деструктор класу*;

Г. *нічого, тому що конструктор за замовчуванням додається в клас автоматично*;

Д. *нічого, тому що деструктор додається в клас автоматично*.

7. Вкажіть вірний варіант доступу до членів об'єктів, описаних наступним чином:

class my {char s; public: double Z; int f (int c, int d) {return c + d;}; } T1, T2;

А. *T1.Z = 23.1*;
Б. *T2-> f (2,1);*
В. *T1.s = '#'*;
Г. *my.T2-> s = 'L'*.

Тема 4 . Успадкування класів, віртуальне успадкування

Ключові поняття: успадкування, базовий клас, похідний клас, множинне успадкування, віртуальне успадкування.

Успадкування – механізм, який дозволяє одним об’єктам отримувати властивості інших об’єктів. Успадкування має на увазі створення одного класу на основі іншого. Клас, на основі якого створюється новий клас, називається **базовим**. Новий клас, що створюється, називається **похідним** класом або **підкласом**.

У загальному випадку існує три типи успадкування:

– **публічний (public)** – публічні (public) і захищені (protected) дані успадковуються без зміни рівня доступу до них;

– **захищений (protected)** – всі успадковані дані стають захищеними;

– **приватний (private)** – всі успадковані дані стають приватними.

На практиці найчастіше використовують *відкрите* (*public*) або *просте* успадкування. Синтаксис успадкування наступний:

class ім’я_базового_класу: [модифікатор_доступа] ім’я_похідного_класу { тіло_класа };

При визначенні похідного класу конструктори не успадковуються – вони створюються в похідному класі (якщо не визначені програмістом явно) **за такими правилами**:

– якщо в базовому класі немає конструкторів, або є конструктор без аргументів (з аргументами за замовчуванням), то в похідному класі конструктор можна не писати – будуть створені конструктори копіювання і конструктор без аргументів;

– якщо в базовому класі всі конструктори з аргументами, похідний клас зобов’язаний мати конструктор, в якому явно повинен бути викликаний конструктор базового класу;

– при створенні об’єкта похідного класу спочатку викликається конструктор базового класу, потім - похідного.

З деструкторами компілятор працює наступним чином:

– при відсутності деструктори в похідному класі система створює деструктор за замовчуванням;

– деструктор базового класу викликається в деструкторі похідного класу автоматично незалежно від того, визначений він явно чи створений системою;

– деструктори викликаються (для знищення об’єктів) в послідовності, зворотній виклику конструкторів.

Приклад:

```
#include <iostream>
#include <string>
```

```
class Person
{
public:
   Person(std::string n, int a)
   {
      name = n; age = a;
   }
   void display()
   {
      std::cout << "Name: " << name << "\tAge: " << age <<std::endl;
   }
private:
   std::string name;
   int age;
};
class Employee : public Person
{
public:
   Employee(std::string n, int a, std::string c):Person(n, a)
   {
      company = c;
      std::cout << "Company" << c <<"  ";
   }
private:
   std::string company;
};

int main()
{
   Person Kate("Kate", 33);
   Kate.display();

   Employee Andrew("Andrew", 33, "ZNU");
   Andrew.display();

   return 0;
}
```

Множинне успадкування – успадкування, при якому один клас успадковує атрибути двох і більше класів одночасно.

Наприклад:

```
class A { int a; };
class B: public A {};
class C: public A {};
class D: public B, public C {};
```

У класі D, в такому випадку, будуть два поля з ім'ям a і вони обидва будуть належати класу A. Проблема полягає у визначенні до якої змінної йде звернення. Для виключення подібної ситуації використовують **віртуальне успадкування**:

```
class A { int a; };
class B: public virtual A {};
class C: public virtual A {};
class D: public B, public C {};
```

? Контрольні запитання:

1. Як ви розумієте поняття «успадкування» у об’єктно-орієнтованому програмуванні ?
2. Які існують типи успадкування у об’єктно-орієнтованому програмуванні?
3. За якими правилами створюються конструктори у похідних класах при успадкуванні?
4. За якими правилами створюються деструктори у похідних класах при успадкуванні?
5. В якому випадку необхідне віртуальне успадкування?

✍Практичні завдання:

1 Створити базовий клас **Car** (машина), що характеризується торговою маркою (рядок), числом циліндрів, потужністю. Визначити методи призначення та змінення потужності. Створити похідний клас **Lorry** (вантажівка) який характеризується також вантажопідйомністю кузова. Визначити функції призначення марки та зміни вантажопідйомності.

2. Створити клас **Pair** (пара чисел); визначити методи зміни полів і порівняння пар: пара p1 більше пари p2, якщо pl.first > p2.first або p1.first = p2.first й pl.second > p2.second. Визначити похідний клас **Fraction** з полями: ціла та дробова частина числа. Визначити повний набір методів порівняння.

3. Створити клас **Liquid** (рідина), що має поля назви і щільності. Визначити методи перепризначення і зміни щільності. Створити похідний клас **Alcohol** (спирт), що має міцність. Визначити методи отримання та зміни міцності.

4. Створити клас **Pair** (пара чисел); визначити методи зміни полів і обчислення добутку чисел. Визначити похідний клас **Rectangle** (прямокутник) з полями-сторонами. Визначити методи обчислення периметра і площі прямокутника.

5. Створити клас **Man** (людина), з полями: ім'я, вік, стать і вага. Визначити методи перепризначення імені, зміни віку і зміни ваги. Створити похідний клас **Student**, що має поле року навчання. Визначити методи перепризначення та збільшення року навчання.

6. Створити клас **Triad** (трійка чисел) й визначити методи зміни полів і обчислення суми чисел. Визначити похідний клас **Triangle** з полями-сторонами. Визначити методи обчислення кутів і площі трикутника.

7. Створити клас **Triangle** з полями-сторонами. Визначити методи зміни сторін, обчислення кутів, обчислення периметра. Створити похідний клас **Equilateral** (рівносторонній), що має поле площі. Визначити метод обчислення площі.

8. Створити клас **Triangle** з полями-сторонами. Визначити методи зміни сторін, обчислення кутів, обчислення периметра. Створити похідний клас **RightAngled** (прямокутний), що має поле площі. Визначити метод обчислення площі.

9. Створити клас **Pair** (пара чисел); визначити методи зміни полів і обчислення добутку чисел. Визначити похідний клас **RightAngled** з полями-катетами. Визначити методи обчислення гіпотенузи і площі трикутника.

10. Створити клас **Triad** (трійка чисел) та визначити метод порівняння **Triad** (див. завдання 2). Визначити похідний клас **Date** з полями: рік, місяць і день. Визначити повний набір методів порівняння дат.

11. Створити клас **Triad** (трійка чисел) й визначити метод порівняння тріад (див. завдання 2). Визначити похідний клас **Time** з полями: година, хвилина і секунда. Визначити повний набір методів порівняння моментів часу.

12. Реалізувати клас **Number** для числового типу float. Реалізувати методи додавання і ділення. Створити похідний клас **Real**, в якому реалізувати метод піднесення до довільного ступеня та метод для обчислення логарифма числа.

13. Створити клас **Triad** (трійка чисел) й визначити методи збільшення полів на 1. Визначити похідний клас **Date** з полями: рік, місяць і день. Перевизначити методи збільшення полів на 1 і визначити метод збільшення дати на n днів.

14. Реалізувати клас-оболонку **Number** для числового типу double. Реалізувати методи множення і віднімання. Створити похідний клас **Real**, в якому реалізувати метод, що обчислює корінь довільного ступеня та метод для обчислення числа π в цьому ступені.

15. Створити клас **Triad** (трійка чисел); визначити методи збільшення полів на 1. Визначити клас-спадкоємець **Time** з полями: година, хвилина, секунда. Перевизначити методи збільшення полів на 1 і визначити методи збільшення на n секунд і хвилин.

16. Створити базовий клас **Pair** (пара цілих чисел) з операціями перевірки на рівність та перемноження полів. Реалізувати операцію віднімання пар по формулі(a, b) -(c, d) = (a - b, c - d). Створити похідний клас **Rational** і визначити нові операції складання(a, b) + (c, d) = (ad + be, bd) і ділення (a, b) / (c, d) = (ad, be). Перевизначити операцію різниці (a, b) -(e, d) = (ad - be, bd).

17. Створити клас **Pair** (пара чисел) та визначити метод множення і операцію додавання пар (a, b) + (c, d) = (a + b, c + d). Визначити похідний

клас **Complex** з полями: дійсна і уявна частині числа. Визначити методи добутку (a, b) x(c, d) = (ac - bd, ad + be) та різниці(a, b) - (c, d) = (a - b, c - d).

18. Створити клас **Pair** (пара цілих чисел) й визначити методи зміни полів і операцію складання пар (a, b) + (c, d) = (a + b, c + d). Визначити клас-спадкоємець **Long** з полями: старша частина числа і молодша частина числа. Перевизначити операцію складання і визначити методи добутку та різниці.

19. Створити базовий клас **Triad** (трійка чисел) та визначити операції складання з числом, множення на число, перевірки на рівність. Створити похідний клас **Vector3D**, що задається трійкою координат. Мають бути реалізовані: операції складання векторів, скалярний добуток векторів.

20. Створити клас **Pair** (пара цілих чисел); визначити метод множення на число і операцію додавання пар (a, b) + (c, d) = (a + b, c + d). Визначити клас-спадкоємець **Money** з полями: рублі й копійки. Перевизначити операцію додавання і визначити методи різниці та ділення грошових сум.

Тести з теми «Успавдкування класів, віртуальне успадкування»

1. Що унаслідується від базового класу в C++. Оберіть усі варіанти.

А. *конструктори;*
Б. *деструктори;*
В. *методи;*
Г. *змінні.*

2. Які твердження для private наслідування є дійсними? Оберіть усі варіанти.

А. *усі public члени базового класу стають private членами похідного класу;*
Б. *protected члени базового класу не доступні у похідному класі;*
В.*усі члени базового класу стають private членами похідного класу.*

3. Оберіть вірний програмний код задання базового класу Parent та похідного від нього класу Child з публічним наслідуванням.

А. *class Parent { public int z;}; class Child:protected Parent{};*
Б. *class Parent { protected int z;}; class Child:public Parent{};*
В. *class Parent { public int z;}; class Child:private Parent{};*

4. Серед перелічених варіантів виберіть модифікатори режиму доступу, які дозволяють змінити тип доступу членам похідного класу до компонентів базового класу.

А. *private;*
Б. *public*;
В. *protected*;
Г. *virtual;*
Д. *base.*

5. Які помилки існують у наступному програмному коді?

```
class A {
public:
A(int x) : _x(x) {} //1
private:
int _x;
};
int main() {
A a = 10; //2
A b(a);
return 0;}
```

А. *у рядку 2 буде помилка, тому що клас А не має конструктора за замовчуванням;*

Б. *у рядку 2 буде помилка, тому що у класі А не оголошений конструктор копіювання за замовчуванням*;

В. *у рядку 2 буде помилка, тому що клас А не має публічного (public) конструктора за замовчуванням;*

Г. *помилок немає.*

Тема 5 . Віртуальні функції, абстрактне успадкування

Ключові поняття: зв'язування, віртуальні функції, перевизначена функція, абстрактний клас.

Віртуальна функція – це функція, оголошена з ключовим словом **virtual** в базовому класі і перевизначена в одному або в декількох похідних класах (у похідних класах вказувати специфікатор virtual не обов'язково).

Перевизначена функція – віртуальна функція, яка перевизначена у похідному класі.

Приклад:

```
#include <iostream>
using namespace std;
class quadrangle
{

public:

  virtual void square( ) {  }
};
class quadrate : public quadrangle
{
public:

  void square( int a) { cout <<"Square of  quadrate is: : " << a*a<<endl; }
};

class rectangle : public quadrangle
{
public:

  void square( int a, int b) { cout <<"Square of  rectangle is: : " << a*a<<endl; }
};
int main()
{
 quadrate c;
c.square(5);
 rectangle b;
 b.square(3,4);
  return 0;
}
```

Функція *square( )*, визначена у базовому класі як *virtual*, у похідних класах викликається в залежності від того, до якого класу належить оголошений екземпляр. Для змінної *c* (екземпляру класу *quadrate*) виконується функція *void square(int a),* яка обчислює площу фігури за одним параметром, а для змінної *в* – функція *void square(int a, int b).*

На основі використання похідних класів і віртуальних функцій реалізується один з принципів об’єктно-орієнтованого програмування – поліморфізм.

Якщо віртуальну функцію у базовому класі зробити чистою (вказати «=0»), то він стане абстрактним:

```
class Box {
public:
virtual double getVolume() = 0;
private:
double length;
double height; };
```

Абстрактний клас – клас, який служить інтерфейсом доступу до похідних від нього класів.

На абстрактні класи накладуються певні обмеження:

– неможливо створити об'єкт абстрактного класу;

– абстрактний клас не можна вживати для завдання типу параметру функції або типу значення, що повертається функцією;

– абстрактний клас не можна використовувати при явному приведення типів; в той же час можна визначити покажчики і посилання на абстрактний клас.

? Контрольні запитання:

1. Що собою являє віртуальна функція?
2. Який принцип об’єктно-орієнтованого програмування реалізується використанням віртуальних функцій?
3. У якому класі перевизначається віртуальна функція?
4. Яким чином визначається яка саме перевизначена віртуальна функція і з якого класу буде виконуватись для певного екземпляру класу?
5. Як визначається абстрактний клас у С++?
6. Які обмеження накладуються на абстрактні класи?

✍Практичні завдання:

1. Створити базовий клас **Figure** з віртуальними методами обчислення площі та периметра. Створити похідні класи **Rectangle**(прямокутний), **Circle**(коло), **Trapezium**(трапеція) зі своїми функціями площі та периметра. Самостійно визначити : які поля необхідні, які з них можна задати в базовому класі, а які – в похідних. Площа трапеції : S = (a + b) x h / 2.

2. Створити базовий клас **Number** з віртуальними методами – арифметичними операціями. Створити похідні класи **Integer** (ціле) та **Real** (дійсне).

3. Створити абстрактний базовий клас **Body**(тіло) з віртуальними функціями обчислення площі поверхні та об'єму. Створити похідні класи : **Parallelepiped** (паралелепіпед) и **Ball**(куля) зі своїми функціями площі поверхні та об'єму.

4. Створити абстрактний клас **Currency**(валюта) для роботи з грошовими сумами. Визначити віртуальні функції переводу в рублі та виводу на екран. Реалізувати похідні класи **Dollar**(долар) і **Euro**(євро) зі своїми функціями переводу та виводу на екран.

5. Створити абстрактний базовий клас **Triangle** для подання трикутника з віртуальними функціями обчислення площі та периметра. Поля даних повинні включати дві сторони та кут між ними. Визначити похідні класи :прямокутний трикутник, рівнобедрений трикутник, рівносторонній трикутник зі своїми функціями обчислення площі та периметра.

6. Створити абстрактний клас **Pair** з віртуальними арифметичними операціями . Створити похідні класи **FazzyNumber** та **Complex**.

Клас **FazzyNumber** для роботи з нечіткими числами, які представляються трійками чисел (x – e1, x, x + e2*)*. Для чисел *A* = (A - *al*, *A, A + ar)* і *B = (B - bl, B, B + br)* арифметичні операції виконуються за наступними формулами:

- додавання A+B = (A + B - al - bl, A + B, A + B + ar + br);
- віднімання A-B = (A-B - al - bl, A - B, A - B + ar + br);
- множення A*B = (A*B – B*al – A*bl + al* bl, A*B, A*B + B* al + A*bl + al*bl);
- обернене число A = (1 / (A + ar), 1/ A, 1 / (A-A- al), A > 0;
- ділення A / B = ((A - al) / (B + br), A / B, (A + ar) / (B - bl)), B > 0;

Клас **Complex** для роботи з комплексними числами. Обов'язково повинні бути присутнім операції:

- додавання add, (a, b) + (e, d) = (a + b, e + d);
- віднімання sub, (a, b) - (e, d) = (a - b, e - d);
- множення mul, (a, b) * (e, d) = (ac - bd, ad + be);
- ділення div, (a, b) / (e, d) = (ac + bd, be - ad) / (e^2 + d^2)
- порівняння equ, (a, b) = (e, d), якщо (a = c) і (b = d);
- спряжене число conj, conj(a, b) = (a, - b).

7. Створити абстрактний базовий клас **Root** (корінь) з віртуальними методами обчислення коренів та виводу результату на екран. Визначити похідні класи: **Linear** (лінійне рівняння) і **Square** (квадратне рівняння) з власними методами обчислення коренів та виводу на екран.

8. Створити абстрактний базовий клас **Function** (функція) з віртуальними методами обчислення значення функції y = f(x) в заданій точці x і виводу результату на екран. Визначити похідні класи : **Ellipse** (еліпс), **Hyperbola** (гіпербола) з власними функціями обчислення y, залежно від вхідного параметру x. Рівняння еліпса $x^2 / a^2 + y^2 / b^2 = 1$; гіперболи: $x^2 / a^2 - y^2 / b^2 = 1$.

9. Створити абстрактний базовий клас **Pair** з віртуальними арифметичними операціями. Реалізувати похідні класи : **Complex** (вар.6) і **Rational**.

Клас Rational для роботи з раціональними дробами. Обов'язково повинні бути реалізовані операції:

- додавання add, (a, b) + (e, d) = (ad + be, bd);
- віднімання sub, (a, b) - (e, d) = (ad - be, bd);
- множення mul, (a, b) x (e, d) - (ac, bd);
- ділення div, (a, b) / (e, d) = (ad, be);
- порівняння equal, greate, less.

10. Створити абстрактний базовий клас **Triad** з віртуальними методами збільшення на 1. Створити похідні класи : **Date** і **Time**.

Клас **Date** для роботи з датами у форматі «рік.місяць.день». Дата представляється структурою із трьома полями типу unsigned int: для року, місяця й дня. Клас повинен включати не менш трьох функцій ініціалізації: числами, рядком виду «рік.місяць.день» (наприклад, «2004.08.31») і датою. Обов'язковими операціями є: обчислення дати через задану кількість днів, вирахування заданої кількості днів з дати, визначення високосного року, присвоєння й одержання окремих частин (рік, місяць, день), порівняння дат (дорівнює, до, після), обчислення кількості днів між датами.

Клас **Time** для роботи з часом у форматі «година:хвилина:секунда». Клас повинен містити в собі не менше чотирьох функцій ініціалізації: числами, рядком (наприклад, «23:59:59»), секундами й часом. Обов'язковими операціями є: обчислення різниці між двома моментами часу в секундах, додавання часу й заданої кількості секунд, вирахування із часу заданої кількості секунд, порівняння моментів часу, переклад у секунди, переклад у хвилини (з округленням до цілої хвилини).

11. Створити абстрактний базовий клас **Pair** з віртуальними арифметичними операціями. Створити похідні класи : **Money** і **Fraction**.

Клас **Money** для роботи із грошовими сумами. Число повинне бути представлене двома полями: типу long для рублів і типу unsigned char - для копійок. Дробова частина (копійки) при виводі на екран повинна бути відділена від цілої частини комою. Реалізувати додавання, віднімання, ділення сум, ділення суми на дробове число, множення на дробове число й операції порівняння.

Клас **Fraction** для роботи із дробовими числами. Число повинне бути представлене двома полями: ціла частина - довге ціле зі знаком, дробова частина - беззнакове коротке ціле. Реалізувати арифметичні операції додавання, віднімання, множення й операції порівняння.

12. Створити абстрактний базовий клас **Pair** з віртуальними арифметичними операціями. Реалізувати похідні класи : **Money** (вар.11) і **Complex** (вар.6) .

13. Створити абстрактний базовий клас **Integer** (ціле) з віртуальними арифметичними операціями і функцією виводу на екран. Визначити похідні

класи : **Decimal** (десяткове) и **Binary** (двійкове),реалізуючи власні арифметичні операції і функцію виводу на екран. Число представляється масивом, кожний елемент якого – цифра.

14. Створити абстрактний базовий клас **Pair** з віртуальними арифметичними операціями. Створити похідні класи : **Money** (вар.11) і **Complex** (вар.6).

15. Створити абстрактний клас **Series** (прогресія) з віртуальними функціями обчислення н-го елемента прогресії та суми прогресії. Визначити похідні класи : **Linear** (арифметична) і **Exponential** (геометрична). (Арифметична прогресія $a_j = a_0 + jd, j = 0,1,2,\ldots$ Сума арифметичної прогресії : $S_n = (n+1)(a_0 + a_n)/2$ Геометрична прогресія: $a_j = a_0 r^j, j = 0,1,2,\ldots$Сума геометричної прогресії: $S_n = (a_0 - a_n r)/(1-r)$

16. Створити абстрактний клас **Norm** з віртуальною функцією обчислення норми та модуля. Визначити похідні класи : **Complex** (вар.6), **Vector3D**.

Vector3D, що задається трійкою координат. Обов'язково повинні бути реалізовані: додавання й віднімання векторів, скалярний добуток векторів, множення на скаляр, порівняння векторів, обчислення довжини вектора, порівняння довжини векторів.

17. Створити абстрактний базовий клас **Pair** з віртуальними арифметичними операціями. Створити похідні класи : **FazzyNumber** (вар.6) і **Fraction**.

Клас **Fraction** для роботи із дробовими числами. Число повинне бути представлене двома полями: ціла частина - довге ціле зі знаком, дробова частина - беззнакове коротке ціле. Реалізувати арифметичні операції додавання, віднімання, множення й операції порівняння.

18. Створити абстрактний базовий клас **Container** з віртуальними методами **sort()** та поелементною обробкою контейнера **foreach**(). Розробити похідні класи **Bubble** (бульбашка) і **Choice** (вибір). В першому класі сортування реалізується бульбашковим методом, а поелементна обробка полягає у вилученні квадратного кореня. У другому класі сортування реалізується методом вибору, а поелементна обробка – обчислення логарифма.

19. Створити абстрактний базовий клас **Array** з віртуальними методами додавання та по елементної обробки масиву **foreach()**. Розробити похідні класи : **SortArray** і **XorArray**. В першому класі операція додавання реалізується як об’єднання множин, а поелементна обробка - як сортування. У другому класі операція додавання реалізується як виключне або, а поелементна обробка - обчислення кореня.

20. Створити абстрактний базовий клас **Array** з віртуальними методами додавання та поелементної обробки масиву **foreach()**. Розробити похідні класи **AndArray** і **OrArray** (вибір). У першому класі операція додавання реалізується як перетин множин, а в другому поелементна обробка представляє собою вилучення квадратного кореня. В другому класі операція

додавання реалізується як об’єднання, а поелементна обробка - як обчислення логарифма.

Тести з теми «Віртуальні функції, абстрактне успадкування»

1. Де обов’язково повинен бути специфікатор virtual при визначенні віртуальної функції?

А. у похідному класі ;

Б. у базовому класі;

В. при оголошенні прототипу функції;

Г. у функції main().

2. Який принцип об’єктно-орієнтованого програмування реалізується використанням віртуальних функцій?

А. поліморфізм;

Б. наслідування;

В. інкапсуляція;

Г. реалізуються усі принципи.

3. У якому класі перевизначається віртуальна функція?

А. у похідному класі ;

Б. у базовому класі;

В. в будь-якому класі;

Г. у абстрактному класі.

4. У якому класі перевизначається віртуальна функція?

А. у похідному класі ;

Б. у базовому класі;

В. в будь-якому класі;

Г. у абстрактному класі.

5. Скільки параметрів може приймати перевизначена віртуальна функція у похідному класі?

А. стільки ж, скільки й у базовому ;

Б. будь-яку кількість параметрів (необхідну для реалізації поставленої задачі);

В. не більше одного;

Г. два або три параметри.

Тема 6. Перевантаження операцій

Ключові поняття: оператор**,** перевантаження операторів, унарний оператор, бінарний оператор

Оператор в С ++ – це деяка дія або функція позначена спеціальним символом. Для того, щоб поширювати ці дії на нові типи даних, при цьому зберігаючи природний синтаксис, в С ++ з'явилась можливість перевантаження операторів.

Перевантаження операторів – один із способів реалізації поліморфізму, що полягає в можливості одночасного існування в одній області видимості декількох різних варіантів застосування оператора, що мають одне і те ж ім'я, але розрізняються типами параметрів, до яких вони застосовуються.

Так, наприклад, функцію sqrt() вже перевантажено для цілого та дійсного типів даних у мові С++. Якщо визвати sqrt(1.5), буде викликано функцію добування кореня для чисел з плаваючою крапкою, а при виклику sqrt(7) – для цілих чисел. Компілятор сам визначає яку функцію вибрати в залежності від сігнатури.

У мові програмування С++ для перевантаження операторів користувацьких типів даних використовується специфікатор **operator.**

Існує можливість перевантаження більш ніж 40 операцій.

При перевантаженні операції потрібно враховувати декілька обмежень.

1. Заборонено перевантаження наступних операцій:
- **sizeof ()** – визначення розміру аргументу;
- **.** (крапка) – селектор компонента об'єкта;
- **? :** – умовна операція;
- **::** – розширення області видимості;
- **.*** – вибір компонента класу через покажчик;
- # та ## – операції препроцесора.

2. Операції можна перевантажувати тільки для нового типу даних (у С++ новий тип даних можна утворити за допомогою конструкцій **enum**, **union** , **struct** та **class**).

3. Не можна змінити пріоритет операції і кількість операндів. Єдина операція , яка не має фіксованої кількості операндів – це операція виклику функції **( ).** Операції «+ » , «-» , « * » , « & » допускається перевантажувати і як унарні , і як бінарні.

4. Не дозволяється використовувати параметри за замовчуванням.

5. Операції можна перевантажувати або як незалежні зовнішні функції (тільки такий спосіб перевантаження допустимий для **enum**), або як методи класу.

6. Чотири операції:
- присвоєння = ;
- виклику функції **();**

– індексування **[ ]** ;

– доступу за вказівником -> ;

допускається перевантажувати тільки як методи класу(ці операції не можна перевантажити для конструкції **enum**).

Унарний оператор – це оператор от одного параметру (++, -- і т.д.)

Приклад перевантаження унарного оператору:

```
#include <iostream>
using namespace std;
class Counter
{
private:
	int count;
public:
	Counter(): count(0)
	{}
	Counter(int c) : count(c)
	{
	}
	int getCount()
	{
		return count;
	}
	Counter operator++() // перевантаження оператору збільшення на 1, для унарных операцій аргументи не вказуються
	{
		return Counter(++count);
	}
};
int main()
{
	setlocale(0, "");
	Counter c1;
	Counter c2;
	cout << c1.getCount() << endl;
	c1++;
	c2 = ++c1;
	cout << c2.getCount() << endl;
	cout << c1.getCount() << endl;
	system("pause");
	return 0;
}
```

Бінарний оператор – це функція від двох параметрів, параметрами якої є лівий і правий операнди оператору.

Приклад перевантаження бінарного оператору:

```
#include <iostream>
using namespace std;
class Time
{
private:
	int hou;
	int min;
	int sec;

public:
	Time() : hou(0), min(0),sec(0)
	{}
	Time(int h, int m, int s) : hou(h), min(m), sec(s)
	{}
	void showTime()
	{
		cout << hou << ":" << min << ":" << sec << endl;
	}
	// t3=t1+t2
	Time operator+ (Time t2) /перевантаження оператору знаходження суми, для бінарных операцій вказується правий операнд
	{
		Time t_res;
		t_res.sec = sec + t2.sec;
		if (t_res.sec >= 60)
		{
		t_res.sec -=60;
		t_res.min = 1;
		}
		t_res.min += min + t2.min;
		if (t_res.min >= 60)
		{
		t_res.min -= 60;
		t_res.hou = 1;
		}
	t_res.hou += hou + t2.hou;
	return t_res;
	}

};
int main()
{
	setlocale(0, "");
	Time t1(2,34,56);
```

```
    Time t2(1,12,5);
    Time t3;
    t3 = t1 + t2;
    t3.showTime();
    system("pause");
    return 0;
}
```

❓ Контрольні запитання:

1. Що таке оператор?
2. З якою метою було визначено перевантаження операторів у С++?
3. Як здійснюється перевантаження операторів у С++?
4. Чим відрізняється перевантаження унарного оператору від перевантаження бінарного оператору?
5. Які операції заборонено перевантажувати у С++?
6. Чи можна перевантажити операції складання та віднімання для змінних типу int?

✍Практичні завдання:

Виконати практичні завдання теми №3 та для об’єктів кожного класу перевантажити унарну операцію ++ та бінарну операцію +.

Тести з теми «Перевантаження операцій»

1. Оберіть вірне перевантаження інкременту для об’єктів класу

```
class numbers
{
private:
        int number;}

A. numbers operator++()        {
                return Numbers(++number);}
;
Б. numbers operator++()        {
                return Numbers(--number);}
;
В. number operator++(){
                return number(++number);}
;
Г. number operator++() {
                return number(--number);}
```

2. Який принцип об’єктно-орієнтованого програмування реалізується при перевантаження операцій?

А. поліморфізм;

Б. наслідування;

В. інкапсуляція;

Г. реалізуються усі принципи.

3. Задано клас

class Strings

private:

{ char str[256];}.

Яку строку необхідно додати у класі, щоб у головній програмі можна було склеїти дві строки типу Strings?

А. Strings operator +(Strings s)
{
strcat(str, s.str);
};

Б. Strings operator +(char s)
{
strcat(str, s);
};

В. Strings operator +(Strings s)
{
strcat(Strings.str, s);
};

Г. Strings operator +(Strings str)
{
strcat(str, s);
};

4. Вкажіть вірну відповідь на питання: які оператори неможна перевантажувати у С++?

А. «sizeof ()»; «.» (крапка); «? :»; «::»; «.»; «#» та «##»;*

Б. «+»; «-»; «? :»; «::»; «.»; «#» та «##»;;*

В. «sizeof ()»; «==»; «? :»; «::»; «.»; «#» та «##»;;*

Г. «sizeof ()»; «+=»; «? :»; «::»; «.»; «#» та «##»;.*

5. Для якого типу даних можна перевантажувати операції?

А. для стандартних вбудованих ;

Б. для користувацьких;

В. для нестандартних.

Тема 7. Обробка виключних ситуацій

Ключові поняття: виключна ситуація, приклади виключних ситуацій, різновиди виключних ситуацій, синхронні та асинхронні виключення, обробка виключних ситуації, блоки try, catch і throw.

Виключна ситуація – це така ситуація, в результаті якої генерується помилка, і виконання програми переривається.

Приклади виключних ситуацій:

– ділення на нуль, вилучення кореня з від'ємного значення;

– помилка при спробі вважати дані з зовнішнього пристрою;

– вичерпання доступної пам'яті;

– поява сигналу аварійного відключення електроживлення системи.

Різновиди виключних ситуацій:

– **синхронні виключення** можуть виникнути тільки в певних, заздалегідь відомих точках програми. Наприклад, помилка читання файлу або комунікаційного каналу, нестача пам'яті;

– **асинхронні виключення** можуть виникати в будь-який момент часу і не залежать від того, яку конкретно інструкцію програми виконує система. Нарпиклад, аварійний відмова живлення або надходження нових даних.

Обробка виключної ситуації – механізм мови програмування, призначений для опису реакції програми на виключні ситуації, які можуть виникнути при виконанні програми і призводять до неможливості (безглуздості) подальшоого відпрацювання програмою її базового алгоритму.

Для обробки виключних ситуацій у C++ використовуються специфікатори: *try, catch і throw*.

У блок *try* розміщуються ті інструкції програми, де очікується можливість появи виключних ситуацій.

У блоці *catch* здійснюється обробка помилки.

Оператор *throw* використовується, щоб сигналізувати про виникнення виключення або помилки. Сигналізування про те, що сталося виключення, називається генерацією виключення (або ще «викиданням виключення»).

СИнтакисис блоків *try і catch*:

```
try {
// блок try
catch (тип1 аргумент) {
// блок catch
catch (тип2 аргумент) {
// блок catch
catch (тип3 аргумент) {
// блок catch
}
```

```
...
catch (типN аргумент) {
// блок catch
}
```

Коли виключення згенеровано, воно перехоплюється відповідною інструкцією catch, яка обробляє цей виняток. Одному блоку try може відповідати кілька інструкцій catch. Яка саме інструкція catch виповнюється, залежить від типу виключення. Це означає, що якщо тип даних, зазначених в інструкції catch, відповідає типу даних виключення, то тільки ця інструкція catch і буде виконана. Коли виняток перехоплено, try отримує її значення.

Загальна форма запису інструкції *throw* має вигляд:

throw виключення;

Інструкція *throw* повинна виконуватися або всередині блоку *try*, або в функції, викликаної з блоку *try*.

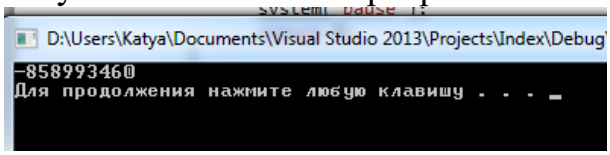
Якщо генерується виключення, для якого відсутня відповідна інструкція *catch*, може статися аварійне завершення програми.

Приклад:

```
#include <iostream>
#include<fstream>
#include <string>
using namespace std;
const int MAX = 3;

class Array
{
private:
	int arr[MAX];
public:

	Array(int *a)
	{
		for (int i = 0; i < MAX; i++)
			arr[i] = a[i];
	}
	int getElement(int index);
};

int Array::getElement(int index)
{
	return arr[index];
}
int main()
{
	setlocale(0, "");
```

```
        int a[] = { 1, 2};
        Array arrayObj(a);
        cout << arrayObj.getElement(2) << endl;
        system("pause");
        return 0;
}
```

Результат виконання програми:



Оскільки у програмі задано виведення не існуючого елементу масиву (у С++ індексація масивів починається з 0), виводиться «мусор».

Щоб подібна ситуація не виникала, необхідно згенерувати обробку виключної ситуації:

```
#include <iostream>
#include<fstream>
#include <string>
using namespace std;
const int MAX = 3;

class Array
{
private:
        int arr[MAX];
public:
        class Extantion {};
        Array(int *a, int len)
        {
                if (len != MAX)
                        throw Extantion();
                for (int i = 0; i < MAX; i++)
                        arr[i] = a[i];
        }
        int getElement(int index);
};

int Array::getElement(int index)
{
        return arr[index];
}
int main()
{
```

```
	setlocale(0, "");
	int a[] = {1, 2};
	try
	{
		Array arrayObj(a, sizeof(a) / sizeof(int));
		cout << arrayObj.getElement(2) << endl;
	}
	catch (Array::Extantion)
	{
		cout << "Помилка! Звернення до не існуючого елементу ";
	}
	system("pause");
	return 0;
}
```

Також можна уточнити яка саме помилка:

```
#include <iostream>
#include<fstream>
#include <string>
using namespace std;
const int MAX = 3;

class Array
{
private:
	int arr[MAX];
public:
	class Big {};
	class Small {};
	Array(int *a, int len)
	{
		if (len > MAX)
			throw Big();
		if (len < MAX)
			throw Small();
		for (int i = 0; i < MAX; i++)
			arr[i] = a[i];
	}
	int getElement(int index);
};

int Array::getElement(int index)
{
	return arr[index];
}
```

```
int main()
{
	setlocale(0, "");
	int a[] = {1, 2};
	try
	{
		Array arrayObj(a, sizeof(a) / sizeof(int));
		cout << arrayObj.getElement(2) << endl;
	}
	catch (Array::Big)
	{
		cout << "Задано занадто великий розмір масиву! "<< endl;
	}
	catch (Array::Small)
	{
		cout << "Задано занадто маленький размір масиву! " << endl;
	}
	system("pause");
	return 0;
}
```

❓ Контрольні запитання:

1. Коли виникає виключна ситуація?
2. Які різновиди виключних ситуацій існують?
3. Як здійснюється обробка виключних ситуацій у C++?
4. Скільки разів можна генерувати блок *try*?
5. Яку функцію виконує оператор *throw* при обробці виключних ситуацій?

✍Практичні завдання:

Реалізувати обробку виключної ситуації, яка може виникнути у задачі.

1. Функція обчислює площу трикутника за трьома сторонами: $S = \sqrt{p(p-a)(p-b)(p-c)}$, де $p = (a+b+c)/2$.
2. Функція обчислює корінь лінійного рівняння $ax + b = 0$.
3. Функція обчислює периметр трикутника.
4. Функція перетворює години та хвилини в секунди.
5. Функція обчислює корінь квадратного рівняння $ax^2 + bx + c = 0$.
6. Функція обчислює суму геометричної прогресії $S_n = (a_0 - a_n r)/(1-r)$.
7. Функція обчислює цілу частину неправильного дробу, що представлений чисельником та знаменником — цілими числами.
8. Функція перетворює комплексне число $z = x + iy$ з алгебраїчної форми в тригонометричну $z = radius(\cos(angle) + i\sin(angle))$. Комплексне число z

представлене структурою. Вихідне число також представити структурою-парою (radius, angle): $radius = \sqrt{x^2 + y^2}$; $angle = arcsin\frac{y}{x}$.

9. Функція обчислює різницю між двома датами в днях. Дати представлені структурою з трьома полями: рік, місяць, день.

10. Функція обчислює кути прямокутного трикутника. В якості параметрів передаються катети A і B. (Синус кута A1, протилежному катету A, обчислюється за формулою $\sin(A1) = a/c$, де c — гіпотенуза трикутника.

11. Функція перевіряє, чи є рядок, який передають, паліндромом.

12. Функція визначає, чи існують прямі $A_1x + B_1x + C_1 = 0$ та $A_2x + B_2x + C_2 = 0$, якщо вираз $d = A_1B_2 - A_2B_1$ не дорівнює нулю. Прямі задаються структурою з трьома полями.

13. Функція обчислює відстань між двома точками $P_1(x_1,y_1)$ та $P_2(x_2,y_2)$ за формулою $D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Виключення генерується, коли P_1 і P_2 — одна й та сама точка.

14. Функція обчислює відстань від точки $P(x_1,y_1)$ до прямої $Ax + Bx + c = 0$ за формулою $D = \frac{|Ax_1 + By_1 + C|}{\sqrt{A^2 + B^2}}$.

15. Функція визначає, чи є рік високосним. Високосність визначається наступним чином: якщо номер року не ділиться на 100, то високосним вважається той рік, який ділиться на 4 без залишку; якщо номер року ділиться на 100, то номер високосного року ділиться на 400 без залишку.

Тести з теми « Обробка виключних ситуацій»

1. Оберіть вірне визначення поняттю «виключна ситуація»

А. це така ситуація, в якій виникає помилка, і виконання програми переривається;

Б. це така ситуація, яка викликає помилку, і виконання програми переривається;

В. це така ситуація, в результаті якої обробляється помилка, і виконання програми переривається

Г. це така ситуація, в результаті якої обробляється помилка, і виконання програми не переривається.

2. Які існують різновиди виключних ситуацій?

А. синхронні та асинхронні;

Б. ділення на нуль, вилучення кореня з від'ємного значення;

В. помилка при спробі вважати дані з зовнішнього пристрою;

Г. вичерпання доступної пам'яті.

3. Який блок здійснює обробку виключної ситуації?

А. catch;

Б. try;

B. throw.

4. Яку строку необхідно додати у класі, щоб у головній програмі можна було склеїти дві строки типу Strings?

A. Strings operator +(Strings s)
{
strcat(str, s.str);
};
Б. Strings operator +(char s)
{
strcat(str, s);
};
В. Strings operator +(Strings s)
{
strcat(Strings.str, s);
};

Г. Strings operator +(Strings str)
{
strcat(str, s);
};

4. Оберіть вірний програмний код обробки ситуації виключення на мові С++.

A.

```
int main()

{
int num1;
int num2;

cout << "Введіть число num1: ";
 cin >> num1;
 cout << "Введіть число num2: ";
 cin >> num2;

 try
 {
 if (num2 == 0)
 {
 throw 999;
 }
 cout << num1 / num2 << endl;
 }
```

```
 catch (int thr)
 {
 cout << "Помилка №" << thr << " - ділення на 0!!!" << endl;
 }
 return 0;
}
;
```

Б.

```
int main()
{
int num1;
int num2;
cout << "Введіть число num1: ";
cin >> num1;
 cout << "Введіть число num2: ";
 cin >> num2;
 catch {
 if (num2 == 0)
 {
 throw 999;
 }
 cout << num1 / num2 << endl;
 }

try (int thr)
 {
 cout << "Помилка №" << thr << " - ділення на 0!!!" << endl;
 }
 return 0;
}
;
```

В.

```
 int main()
{
int num1;
int num2;

cout << "Введіть число num1: ";
 cin >> num1;
 cout << "Введіть число num2: ";
 cin >> num2;

 throw
 {
 if (num2 == 0)
```

```
 {
try 999;
 }
 cout << num1 / num2 << endl;
 }

 catch (int thr)
 {
 cout << "Помилка №" << thr << " - ділення на 0!!!" << endl;
 }
 return 0;
}
```

5. Де повинна виконуватися інструкція *throw*?

А. або всередині блоку try, або в функції, викликаної з блоку try;
Б. або всередині блоку catch, або в функції, викликаної з блоку catch;
В. де завгодно.

Тема 8. Динамічне виділення пам'яті

Ключові поняття: динамічне виділення пам’яті, покажчик,

Перед виконання програми завантажуються у оперативну пам’ять. При статичному оголошенні змінних, вони залишаються в пам'яті до того моменту, як програма завершить свою роботи і знищуються при завершенні програми.

Таке виділення памяті можливе у простих прикладах і нескладних програмах, які не вимагають великої кількості ресурсів. Але при розробці великого проекту з великою кількістю даних виділяти під них пам’ять необхідно більш раціонально. З цієї причини, в С ++ було введено поняття покажчика.

Покажчик – це змінна, що зберігає в собі адресу комірки оперативної пам'яті.

Можна звертатися, наприклад до масиву даних через покажчик, який буде містити адресу початку діапазону комірок пам'яті, що зберігають цей масив. Після того, як цей масив стане не потрібний для виконання решти програми, пам'ять звільняється за адресою цього покажчика. Такий підхід до використання оперативної пам’яті називається – динамічним виділенням пам’яті.

Динамічне виділення пам’яті – спосіб виділення оперативної пам'яті комп'ютера для об'єктів в програмі, при якому виділення пам'яті під об'єкт здійснюється під час виконання програми.

У мові програмування С++ динамічне виділення пам'яті здійснюється за допомогою оператора **new**:

*тип_даних * ім’я_gjrf;xbrf = new тип_даних ;*
наприклад *int *a = new int ;*.

Для того, щоб звільнити пам'ять, виділену оператором **new**, використовується оператор **delete**.

Приклад:

```
#include <iostream>
using namespace std;
int main() { int *a = new int;
int *b = new int;
float *c = new float;
 delete c;
delete b;
delete a;
return 0; }
```

Виділення пам’яті під динамічний масив відбувається також оператором new:

```
#include <iostream>
using namespace std;
int main() {
int num, *p_darr; // розмір масиву
cout << "Enter integer value: ";
cin >> num;
p_darr=new int[num]; // виділення пам’яті під масив
 for (int i = 0; i < num; i++)
{
p_darr[i] = i;
cout << "Value of " << i << " element is " << p_darr[i] << endl;
}
delete [] p_darr;
return 0;}
```

❓ Контрольні запитання:

1. В чому різниця між статичним та динамічним виділенням пам’яті при отладці програми?
2. На що вказує покажчик при його використанні для задання масиву?
3. За допомогою яких операторів задається динамічне виділення пам’яті у С++?
4. З якою метою використовується оператор delete при динамічному виділенні пам’яті у С++?

✍Практичні завдання:

Написати програму, в якій створюються динамічні масиви і виконати їх обробку.

1. Сформувати одновимірний масив. Видалити з нього елемент із заданим номером, додати елемент із заданим номером.
2. Сформувати одновимірний масив. Видалити з нього елемент із заданим ключем, додати елемент із заданим ключем.
3. Сформувати одновимірний масив. Видалити з нього 10 елементів, починаючи з заданого номера, додати елемент із заданим ключем.

4. Сформувати одновимірний масив. Видалити з нього елемент із заданим номером, додати 10 елементів, починаючи з заданого номеру.

5. Сформувати одновимірний масив. Видалити з нього 10 елементів, починаючи з заданого номера, додати 10 елементів, починаючи з заданого номеру.

6. Сформувати двовимірний масив. Видалити з нього рядок із заданим номером.

7. Сформувати двовимірний масив. Видалити з нього стовпець із заданим номером.

8. Сформувати двовимірний масив. Додати до нього рядок із заданим номером.

9. Сформувати двовимірний масив. Додати до нього стовпець із заданим номером.

10. Сформувати двовимірний масив. Видалити з нього рядок і стовпець із заданим номером.

11. Сформувати двовимірний масив. Додати до нього рядок і стовпець із заданим номером.

12. Сформувати двовимірний масив. Видалити з нього всі рядки, в яких зустрічається задане число.

13. Сформувати двовимірний масив. Видалити з нього всі стовпці, в яких зустрічається задане число.

14. Сформувати двовимірний масив. Видалити з нього рядок і стовпець, на перетині яких знаходиться мінімальний елемент.

15. Сформувати двовимірний масив. Видалити з нього рядок і стовпець, на перетині яких знаходиться максимальний елемент.

Тести з теми «Динамічне виділення пам'яті»

1. З якою метою використовується динамічне виділення пам’яті під данні при розробці програм?

А. з метою раціонального використання оперативної пам’яті під час виконання программи;

Б. з метою надійного зберігання даних під час виконання программи;

В. з метою виділення оперативної пам’яті під данні до початку виконання программи;

2. Які оператори використовуються у C++ для динамічного виділення пам’яті?

А. new, delete;

Б. calloc(), malloc();

В. free(), realloc();

3. На що вказує покажчик p у наступному виділенні оперативної пам’яті int *p = new int[4]?

А. на перший елемент масиву з чотирьох об’єктів типу int;

Б. на масив з чотирьох об’єктів типу int*;*

В. на перший елемент масиву з п’яти об’єктів типу int*.*

4. Оберіть вірний синтаксис для динамічного виділення пам’яті під змінну цілого типу?

*А. int *a = new int;*

Б. int a = new int;

*В. int *a = new.*

5. Оберіть строку яка звільняє оперативну пам’ять з-під масиву оголошеного наступним чином *int num, *p; p=new int[num];*?

А. *delete [] p;*

Б. *delete p[];*

В. *delete p;*

Г. *delete p[num];*

РОЗДІЛ 2 АБСТРАКЦІЯ ДАНИХ. СКЛАДЕНІ СТРУКТУРИ ДАНИХ. ОБ'ЄКТНЕ ПРОГРАМУВАННЯ НА БАЗІ С++

Тема 9. Бібліотека потокових класів С++

Ключові поняття: поток, потік вводу/виводу, стандартний потік виводу, стандартний потік вводу, форматований вивід у С++

Потік – це послідовність символів, до якої можна отримати доступ. Згодом потік може виробляти або споживати потенційно необмежені обсяги даних.

Розрізняють потік вводу та потік виводу.

Потік вводу – потік, який використовується для зберігання даних, отриманих від джерела даних: клавіатури, файлу, мережі і т.д.

Потік виводу – потік, який використовується для зберігання даних, що надаються конкретному споживачеві даних: монітору, файлу, принтеру і т.д. При записі даних на пристрій виводу, цей пристрій може бути не готовим прийняти дані негайно - наприклад, принтер все ще може прогріватися, коли програма вже записує дані в вихідний потік. Таким чином, дані будуть перебувати в потоці виведення до тих пір, поки принтер не почне їх використовувати.

За функціонал вводу/виводу даних у С++ відповідають бібліотеки:

- **iostream** – операції вводу/виводу;
- **fstream –** операції вводу/виводу з файлу.

У мові С++ з кожним пристроєм асоціюється потік вводу/виводу:

- **cin** – стандартний потік виводу (клавіатура)
- **cout** – стандартний потік виводу (термінал)
- **cerr** – стандартний потік помилок.

При роботі з потоками відбувається автоматичний перехід у строковий тип даних при виведенні на екран і зі строкового в чисельний тип – при введенні.

Для роботи зі стандартними потоками підключається заголовочний файл iostream.

Ввод/вивод чисельних типів даних:

```
int iVal;
double dVal;
cin>>iVal;
cin>>dVal;
cout<<iVal<<' '<<dVal<<endl;
```

За замовчуванням потоковий ввод cin вводить рядок до пробілу, символу табуляції або переведення рядка.

```
char s[80];
```

```
cin >> s;
```

Для вводу тексту до символу кінця рядка використовується маніпулятор потоку getline ():

```
char name[256], title[256];
  cin.getline (name,256);
  cout << "Enter your favourite movie: ";
  cin.getline (title,256);
  cout << name << "'s favourite movie is " << title;
  return 0;
```

Основні маніпулятори форматованого виводу:

– **cout.fill('/*symbol*/')** – встановлює символ заповнювач; symbol – символ заповнювач, символ задається у одинарних лапках;

– **cout.width(/*width_field*/)** – задає ширину поля, де width_field – кількість позицій (одна позиція – один символ):

```
  double b = 4.3981;
  cout.fill('0');
  cout.width(10);
  cout << a << endl << b << endl;
```

Результат:

00004.3981

– **cout.precision(/*number*/)** – задає кількість значущих цифр у числі (або після коми) в залежності від використання fixed;

```
double a = -112.234;
double b = 4.3981;
cout.precision(4);
cout << a << endl << b << endl;
```

Результат

-112.2

4.398

– **cout.fixed** – показує, що встановлена точність відноситься до кількості знаків після коми;

```
double a = -112.234;
double b = 4.3981;
cout.precision(4);
 cout << fixed<< endl << a << endl << b << endl;
```

Результат:

-112.2340

4.3981

– **cout.showpos** – показує знак + для додатніх чисел;

```
double b = 4.3981;
cout << showpos << b << endl;
```

Результат:

+4.3981

– **cout.scientific** – виводить число у експоненціальній формі:

```
double a = -112.234;
double b = 4.3981
cout  << scientific << a << endl << b << endl;
```

Результат:

```
-1.122340e+002
4.398100e+000
```

❓ Контрольні запитання:

1. З якою метою було введено використання потоку при вводі/виводі даних у С++?
2. Які бібліотеки С++ відповідають за ввід/вивід даних?
3. Якого типу даних набувають данні при виведенні С++?
4. Які маніпулятори форматованого виводу використовуються у С++?

✍Практичні завдання:

Запрограмувати форматований вивід чисельних результатів практичних завдань теми 7. Необхідно використати усі маніпулятори вводу/виводу, які наведені у темі 9.

Тести з теми «Бібліотека потокових класів С++»

1. Який заготовочний файл підключається у програмі на С++ для роботи зі стандартними потоками?

А. iostream;
Б. stdio;
В. fstream.

2. Вкажіть функції потокового вводу/виводу?

А. scanf, printf;
Б. cout, cin;

3. Результат виводу наступного програмного коду:

```
double x = 222.22;
double y = 44.4444;
cout.precision(4);
cout << fixed<< endl << x << endl << y << endl;
```

?

А.

```
222.2200
44.4444;
```

Б.

```
222.2
44.44;
```

В.

222.2200
44.444400.

4. Оберіть програмний код, який виведе значення 33.333 та 666.666 у наступному форматі
**33.333
*666.666?
А.
```
double a=33.333, b=666.666;
cout.fill('*');
  cout.width(8);
  cout << a << endl << b << endl;
```

Б.
```
double a=33.333, b=666.666;
cout.fill('*');
 cout.width(10);
 cout << a << endl << b << endl;
```

В.
```
double a=33.333, b=666.666;
cout.fill('*');
  cout.width(7);
  cout << a << endl << b << endl;
```

5. Який маніпулятор об’єкту cout необхідно задати, щоб вивести число у експоненціальній формі?
А**.** *cout.scientific;*
Б. *cout.showpos;*
В. *cout.fixed;*
Г. cout.precision.

Тема 10. Робота з текстовими файлами. Маніпулятори вводу-виводу.

Ключові поняття: файл, повне ім’я файлу, fstream, режими відкриття файлу

Файл – це іменований набір байтів, який може бути збережений на деякому накопичувачі.

Повне ім'я файлу – це повна адреса до директорії файлу із зазначенням імені файлу, наприклад: E: \ programms \ lab1.cpp

Для роботи з файлами необхідно підключити заголовочний файл **<fstream>,** у якому підключені заголовки **<ifstream>** – файловий ввід і <**ofstream**> – файловий вивід.

Файлові ввід/вивід аналогічні стандартному вводу/виводу, тільки ввід/вивід виконується не на екран, а в файл. Для організації файлового вводу/ виводу необхідно створити власні об'єкти, які будуть використовуватись аналогічно операторам cin і cout.

Наприклад:

1. Запис до файлу data.txt рядка «Об’єктоно-орієнтоване програмування в С++»

```
#include "stdafx.h"
#include <fstream>
using namespace std;

int main()
{
ofstream fout("data.txt"); // створюється об’єкт класу ofstream і зв’язується з файлом data.txt
fout << "Об’єктно-орієнтоване програмування в С++"; // записується рядок до файлу
fout.close(); // закрывається файл
system("pause");
return 0;
}
```

2. Зчитування рядка «Об’єктоно-орієнтоване програмування в С++» з файлу data.txt

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

int main()
{
setlocale(LC_ALL, "rus");
char buff[50]; // буфер для зчитування тексту з файлу
ifstream fin("data.txt"); // відкривається файл для читання
```

```
fin >> buff; // зчитується перше слово з файлу
cout << buff << endl; // друкується це слово
fin.getline(buff, 50); // зчитується рядок з файлу
fin.close(); // закривається файл
cout << buff << endl; // друкується рядок
system("pause");
return 0;
}
```

Якщо в програму передається ім'я неіснуючого файлу або в імені допущена помилка, тоді компілятор проігнорує рядки, де виконується робота з файлом робота програми коректрно, але нічого, на екрані показано не буде. Для виведення повідомлення о помилці в С ++ передбачена is_open (), яка повертає цілі значення: 1 – якщо файл був успішно відкритий, 0 – якщо файл відкритий не був:

```
if (!fin.is_open()) // якщо файл не відкрито
cout << "Файл не може бути відкрито!\n";
{
fin >> buff;
cout << buff << endl;
fin.getline(buff, 50);
fin.close();
cout << buff << endl;
}
```

Режими відкриття файлу:

Константа	**Опис**
ios_base::in	відктрити файл для читання
ios_base::out	відкрити файл для запису
ios_base::ate	при відкритті переместити покажчик в кінець файлу
ios_base::app	відкрити файл для запису в кінець файлу
ios_base::trunc	вилучити вміст файлу, якщо воно існує
ios_base::binary	відкриття файлу у двійниковому режимі

Режими відкриття файлів можна встановлювати при створенні об'єкту або при виконанні функції open ():

```
ofstream fout("data.txt", ios_base::app);
fout.open("data.txt", ios_base::app);
```

Режими відкриття файлів можна комбінувати за допомогою порозрядної логічної операції або |, наприклад: *ios_base :: out | ios_base :: trunc* – відкриття файлу для запису, попередньо очистивши його.

? Контрольні запитання:

1. Що включає в себе повне ім’я файлу?
2. Що відбувається при наступному об’явлення об’єкту fout ofstream fout("data.txt");?
3. Які існують режими відкриття файлу у С++?
4. Як задаються режими відкриття файлу у С++?

✍Практичні завдання:

Розробити програму, яка буде обчислювати характеристики основних типів даних в С ++ і записувати їх в файл в такому форматі:

Тип даних	Розмір у байтах	Максимальне значення
Bool	1	255.00
Char	1	255.00
short int	2	32767.00
unsigned short int	2	65535.00
int	4	2147483647.00
unsigned int	4	4294967295.00
long int	4	2147483647.00
unsigned long int	4	4294967295.00
float	4	2147483647.00
long float	8	92233720368547758 00.00
double	8	92233720368547758 00.00

Для обчислення розміру типу даних у байтах необхідно використати функцію sizeof().

Тести з теми «Робота з текстовими файлами. Маніпулятори вводу-виводу.»

1. Який заголовочний файл підключається у програмі на С++ для роботи з файлами?

А. iostream;
Б. stdio;
В. fstream.

2. Вкажіть вірний рядок (рядки) для запису тексту до файлу data.txt?

А.
```
ofstream fout("data.txt");
fout << "Об’єктоно-орієнтоване програмування в С++";
```
Б.
```
ifstream fin("data.txt");
fin << "Об’єктоно-орієнтоване програмування в С++";
```
В.

ofstream fin ("data.txt");
fin << "Об'єктоно-орієнтоване програмування в С++";

3. Оберіть вірний синтаксис відкриття файлу для запису.
А.
ofstream fout("data.txt", ios_base::out);
Б.
ofstream fout("data.txt", ios_base::app);
В.
ofstream fout("data.txt", ios_base::in);
4. Оберіть вірний синтаксис відкриття файлу для читання.
А.
ofstream fout("data.txt", ios_base::out);
Б.
ofstream fout("data.txt", ios_base::app);
В.
ofstream fout("data.txt", ios_base::in);

5. Оберіть вірний синтаксис відкриття файлу для запису в кінець файлу.
А.
ofstream fout("data.txt", ios_base::out);
Б.
ofstream fout("data.txt", ios_base::app);
В.
ofstream fout("data.txt", ios_base::in);

Тема 11 . Робота з бінарними файлами. POD-типи даних

Ключові поняття: бінарний файл, POD-тип даних

Бінарний файл – файл, в якому дані представлені у внутрішній формі. А оскільки при внутрішньому поданні використовується двійкова система числення, то й файли називаються двійковими. Двійковий файл є аналогом внутрішньої (оперативної, фізичної) пам'яті – необмеженим масивом байтів з можливістю безпосереднього звернення (довільного доступу) до будь-якої його частини.

Така модель файлу повністю збігається з системою уявлень, прийнятої в С для роботи з пам'яттю на низькому (фізичному рівні).

Фізична пам'ять має байтну структуру – одиницею адресації є байт.

Будь-яка змінна займає фіксовану кількість байтів, яка визначається її типом. Операція sizeof повертає цю розмірність.

Покажчик на змінну – її адреса в пам'яті.

Перетворення типу покажчика до void інтерпретує його як «чиста» адреса, а перетворення до char – як покажчик на масив байтів (фізичне уявлення пам'яті).

Для того, щоб відкрити бінарний файл, необхідно задати режим доступу ios :: binary (в деяких компіляторах С ++ – ios :: bin). Бінарні файли більш компактні і в деяких випадках більш зручні для обробки. Для створення вихідного файлу створюють об'єкт:

```
ofstream out_fil (”Outfil.dat”,ios::out | ios::binary);
if (! out_fil) {   cerr<<”Error: Outfil.dat”<<endl;
 exit(1);
}
```

Для того, щоб відкрити існуючий двійковий файл для читання, потрібно створити об'єкт:

```
ifstream in_fil (”Infil.dat”, ios::in | ios::binary);
if (! in_fil) {    cerr<<”Error: Infil.dat”<<endl;
   exit(2);
   }
```

Розглянемо приклад запису значення типу double в бінарний файл:

```
# include <fstream>
# include <iostream>
using namespace std;
# include <stdlib.h>
class bin_outstream:public ofstream
{  public:
   bin_outstream(const char *fn): ofstream(fn, ios::out | ios::binary){}
   void writeOurDate(const void*, int);
   ofstream &operator<<(double d) {   writeOurDate(&d, sizeof(d));
   return *this;
   }
```

```
};
int main()
{   bin_outstream bin_out("B_out.dat");
    if (! bin_out)  {  cerr<<"Unable to write to B_out.dat"<<endl;
    exit(1);
    }
    double d = 5.252;
    bin_out<<d;
    bin_out<<d*d;
    d = 5.2E-5;
    bin_out<<d;
    return 0;
}
void bin_outstream:: writeOurDate(const void *Ptr, int len)
{   if (! Ptr)  return;
    if (len<=0)  return;
    write((char*)Ptr, len);
```

Та приклад читання значень типу double з бінарного файлу.

```
# include <fstream>
# include <iostream>
using namespace std;
# include <stdlib.h>
class bin_instream: public ifstream
{   public:
    bin_instream(const char *fn): ifstream(fn, ios::in | ios::binary){}
    void readOurDate(void*, int);
    bin_instream &operator>>(double &d) {  readOurDate(&d, sizeof(d));
    return *this;
    }
};
int main()
{  bin_instream bin_in("B_in.dat");
    if (! bin_in)
    {  cerr<<"Unable to open B_in.dat"<<endl;
    exit(1);
    }
    double d;
    long count = 0;
    bin_in>>d;
    while (! bin_in.eof())
    {  cout << ++count << ": " << d << endl;
    bin_in>>d;
    }
    return 0;
}
```

```
void bin_instream:: readOurDate(void *p, int len)
{    if (! p) return;
   if (len <= 0) return;
   read((char*)p, len);
}
```

? Контрольні запитання:

1. В чому полягають особливості роботи з двійковими файлами?
2. Що являє собою файловий покажчик?
3. Як організувати доступ до довільного місця виконуваного файлу?

✍Практичні завдання:

1. У бінарному файлі цілого типу замінити максимальний елемент сумою попередніх елементів, мінімальний - сумою наступних елементів.
2. В кінець бінарного файлу цілого типу дописати парні елементи цього файлу.
3. На початок бінарного файлу цілого типу дописати непарні елементи цього файлу.
4. У середину бінарного файлу цілого типу помістити елементи цього файлу, кратні п'яти.
5. У бінарному файлі цілого типу поміняти місцями елементи, що стоять на парних місцях з елементами, що стоять на непарних місцях.
6. На початок бінарного файлу цілого типу дописати його мінімальне значення, в середину - максимальне.
7. На початок бінарного файлу цілого типу записати елементи, які є дільниками максимального елемента цього файлу.
8. У середину бінарного файлу цілого типу записати елементи цього файлу, менші числа, введеного з клавіатури.
9. Дано бінарні файли f і g цілого типу. Записати в початок файлу f позитивні компоненти файлу g, а в кінець файлу g - негативні компоненти файлу f зі збереженням порядку їх слідування.
10. Дано бінарний файл з цілими числами. Видалити з нього число, записане після першого нуля (прийняти, що нулі в файлі є). Результат записати в інший файл.
11. Дано бінарний файл з цілими числами. Всі його парні елементи замінити нулями. Розглянути 2 варіанти: вихідний файл містить 13 чисел; розмір вихідного файлу невідомий.
12. Дано бінарний файл з цілими числами. Замінити всі його елементи, порядковий номер яких кратний 7, на нові значення, які вводяться з клавіатури. Розглянути 2 варіанти: вихідний файл містить 20 чисел; розмір вихідного файлу невідомий.

13. Дано бінарний файл з позитивними і негативними цілими числами. Записати в інший файл спочатку негативні елементи, а потім позитивні.

14. У бінарному файлі цілого типу замінити кожен елемент сумою попередніх елементів. В кінці дописати загальну суму всіх елементів.

15. У кінець бінарного файлу цілого типу дописати все його елементи кратні

Тести з теми «Робота з бінарними файлами. POD-типи даних»

1. Яке з визначень не є вірним для поняття «бінарний файл»?

А.
бінарний файл – файл, в якому дані представлені у внутрішній формі;
Б.
бінарний файл – файл, в якому дані представлені у двійковому вигляді;
В.
бінарний файл – файл, в якому дані представлені у текстовому вигляді.

2. Оберіть вірний синтаксис створення двійкового файлу data.dat.

А.
```
ofstream out_fil (”data.dat”,ios::out | ios::binary);
if (! out_fil)  {   cerr<<”Error: data.dat”<<endl;
  exit(1);
}
```
Б.
```
ofstream out_fil (”data.dat”,ios::out);
if (! out_fil)  {   cerr<<”Error: data.dat”<<endl;
  exit(1);
}
```
В.
```
ofstream out_fil (”data.dat”,ios::out | ios::binary);
if (! out_fil)  {   cout<<”Error: data.dat”<<endl;
  exit(1);
}
```

3. Оберіть вірний синтаксис відкриття двійкового файлу data.dat для читання.

А.
```
ifstream in_fil (”data.dat”, ios::in | ios::binary);
if (! in_fil)  {    cerr<<”Error: data.dat”<<endl;
    exit(2);
    }
```
Б.
```
ifstream in_fil (”Infil.dat”, ios::in | ios::binary);
```

```
if (! in_fil) {    cerr<<”Error: Infil.dat”<<endl;
   exit(2);
   }
```

В.

```
ifstream in_fil (”data.dat”, ios::in);
if (! in_fil) {    cerr<<”Error: data.dat”<<endl;
   exit(2);
   }
```

4. Оберіть вірний синтаксис відкриття двійкового файлу data.dat для запису.

А.

```
ofstream out(“data,dat”,ios::binary|ios::out);
```

Б.

```
ofstream out(“data,dat”,ios::binary|ios::in);
```

В.

```
ofstream out(“data,dat”,ios::base|ios::out).
```

5. У якому рядку програмного коду виведення вмісту двійкового файлу э помилка:

```
#include <iostream.h>
#include <fstream.h>
int main(int argc, char *argv[])
{
string ch;
if (argc!=2) {
cout << "Usage: PR <filename>\n";
return 1;
}
ifstream in(argv[1], ios::in | ios::binary);
if (!in) {
cout << "Cannot open file.\n";
return 1;
}
while (in) {
in.get (ch);
cout << ch;
}
in.close();
return 0;
}?
```

А.

```
string ch;
```

Б.

ifstream in(argv[1], ios::in | ios::binary);

B.

#include <fstream.h>.

Тема 12 . Шаблони функцій та класів. Бібліотеки шаблонів

Ключові поняття: шаблони функцій, синтаксис шаблону у С++, екземпляр шаблону, вимоги до фактичних параметрів шаблону, шаблон класу.

Шаблон (англ. template) – засіб мови С++, призначений для кодування узагальнених алгоритмів, без прив'язки до деяких параметрів (наприклад, типам даних, розмірами буферів, значенням за замовчуванням).

Механізм шаблонів дозволяє відокремити загальний алгоритм від його реалізації стосовно до конкретних типів даних. Використовуваний тип даних є в параметром шаблону.

У мові С ++ є два типи шаблонів – шаблони функцій і шаблони класів.

Шаблони функцій

Оголошення шаблону функції починається з заголовка, що складається з ключового слова template, за яким слідує список параметрів шаблону.

```
// Опис шаблону функції
template <class X>
X sub (X t1, X t2)
{  return (t1-t2);
}
```

Ключове слово class в описі шаблону означає тип, ідентифікатор в списку параметрів шаблону X означає ім'я будь-якого типу.

```
// Використання  шаблону функції
int m = sub (2, 3);
...
// Примірник шаблону функції генерується компілятором
int sub (int t1, int t2)
{  return (t1-t2);
}
```

У списку параметрів шаблону слово class може також відноситься до звичайного типу даних. Таким чином, список параметрів шаблону просто означає, що Т являє собою тип, який буде поставлено пізніше. Так як Т є параметром, що позначає тип, шаблони іноді називають параметризованими типами.

Наведемо опис шаблону функції:

```
template <class T>
T toExcent (T base, int exponent)
{   T res = base;
    if (exp==0) return (T)1;
    if (exp<0) return (T)0;
    while (--exp) res *= base;
    return res;
}
```

Змінна result має тип T. Коли передане в програму значення є 1 або 0, то воно спочатку приводиться до типу T, щоб відповідати оголошенню шаблону функції. Типовий аргумент шаблону функції визначається згідно типам даних, використовуваних у виклику цієї функції: int i = to*Excent* (10, 3); long l = to*Excent* (1000L, 4); double d = to*Excent* (1e5, 5); У першому прикладі T стає типом int, в другому - long. Нарешті, в третьому прикладі T стає типом double. Наступний приклад приведе до помилки компіляції, так як в ньому змінна, яка передається та повертається мають різні типи: int i = to*Excent* (1000L, 4).

Шаблонти класів

Можна створювати шаблони і для класів. Розглянемо приклад – клас, який зберігає декілька значень. Функції-елементи цього класу повертають мінімальне і максимальне значення, а також дозволяють визначити, чи є два значення однаковими:

```
template <class T>
class Two
{   T x, y;
   public:
   Pair (T x1, T y1);
   T Max();
   T Min ();
   int isEqual ();
};
```

Різниця з шаблоном функції в даному випадку полягає в тому, що замість опису функції використовується оголошення класу. Шаблони класів стають більш складними, коли описуються функції, які належать класу. Наприклад, опис належить функції Min () класу Pair:

```
template <class T>
T Pair <T>::Min()
{   return a < b ? a : b;
}
```

Щоб зрозуміти цей запис, давайте повернемося трохи назад. Якби Pair був звичайним класом (а не шаблоном класу) і T був би деяким конкретним типом, то функція Min класу Pair була б описана таким чином:

```
T Pair::Min()
{   return a < b ? a : b;
}
```

Для випадку шаблонної версії необхідно, по-перше, додати заголовок шаблону template. Потім потрібно дати ім'я класу.

! При визначенні шаблону описується безліч класів – сімейство класів.

Описи методів поміщаються в заголовки, так як вони повинна бути видимі всюди, де використовується клас Pair.

```
// конструктор
```

```
template <class T>
Pair <T>::Pair (T t1, T t2) : a(t1), b(t2)
{}
// метод Max
template <class T>
T Pair <T>::Max()
{   return a>b ? a : b;
}
// метод isEqual
template <class T>
int Pair <T>::isEqual()
{   if (a==b) return 1;
return 0;
}
```

? Контрольні запитання:

1. Що таке шаблони і з якою метою вони використовуються?
2. Якого типу шаблони використовуються в програмах?
3. Як оформляються шаблони функцій?
4. Які переваги програми забезпечуються при використанні шаблонів класів?

✍Практичні завдання:

1. Опишіть параметризований клас: однозв'язний список елементів (параметр – тип).
2. Опишіть параметризований клас: двусвязний список елементів (параметр – тип).
3. Опишіть параметризований клас: чергу елементів (параметр – тип).
4. Опишіть параметризований клас: стек елементів (параметр – тип).
2. Опишіть параметризований клас: стек елементів обмеженою ємності (параметр – тип і число).
3. Опишіть параметризований клас: геометрична фігура на площині (параметр – тип і число).
4. Опишіть параметризовану функцію сортування вставкою.
5. Опишіть параметризовану функцію сортування вибіркою.
6. Опишіть параметризовану функцію сортування бульбашкою.
7. Опишіть параметризовану функцію знаходження елемента в неврегульованих масиві.
8. Опишіть параметризовану функцію знаходження елемента в упорядкованому масиві.
9. Опишіть параметризовану функцію заміни одного елемента масиву на інший.
10. Опишіть параметризовану функцію інверсії масиву елементів.

11. Опишіть параметризовану функцію обчислення середнього арифметичного значення масиву елементів.

12. Опишіть параметризований клас: двійкове дерево елементів (параметр – тип).

Тести з теми «Шаблони функцій та класів. Бібліотеки шаблонів»

1. Оберіть найбільш точне визначення поняттю «шаблон» у об'єктно-орієнтованому програмуванні.

А.

шаблон – це засіб об'єктно-орієнтованого програмування, який дозволяє кодувати загальні алгоритми без вказівки типів даних, розмірів структур даних та т.п.;

Б.

шаблон – це засіб об'єктно-орієнтованого програмування, який дозволяє задавати функції без вказівки типів даних, що передаються та тих даних, які функція повертає;

В.

шаблон – це засіб об'єктно-орієнтованого програмування, який дозволяє задавати класи без вказівки типів даних, розмірів структур даних та т.п..

2. Вкажіть вірний синтаксис завдання шаблону функції виводу змінної на еркан.

А.

```
template < typename T >
void ShowValue(T Value)
{
  std::cout << Value;
}
```

Б.

```
typename < template T >
void ShowValue(T Value)
{
  std::cout << Value;
}
```

В.

```
typename < template T >
void ShowValue(T Value)
{
  std::cout << Value;
}
```

3. Чим відрізняється використання функції від використання шаблону функції?

А. в функції тип вхідних та вихідних параметрів задається при її описі, а у шаблоні функції тип параметрів задається при його використанні;

Б. в функції кількість вхідних парметрів задається при її описі, а у шаблоні функції кількість параметрів може змінюватись при його використанні;

В. функція має вхідні та віхідні параметри задаються при її описі , а у шаблон функції зовсім не має параметрів.

4. Оберіть шаблон класу без помилки.

А.

```
template <class T>
class MyNumber
{
   public:
   MyNumber(void) { }
   void Mult2(T* t);
   T MySquare(T);};
```

Б.

```
template <class MyNumber >
class MyNumber
{
   public:
   MyNumber(void) { }
   void Mult2(T* t);
   T MySquare(T);};
```

В.

```
template <class T>
class T
{
   public:
   MyNumber(void) { }
   void Mult2(T* t);
   T MySquare(T);};
```

5. Оберіть вірний синтаксис опису функції Sum шаблону класу Math

А.

```
template <class T>
T Math <T>::Sum()
{   return a+ b;
}
```

Б.

```
template <class T>
```

```
T Math::Sum()
{   return a+ b;
}
```

B.

```
template <class Math>
T Math <T>::Sum()
{   return a+ b;
}
```

Тема 13. Стандартна бібліотека STL: контейнерні класи

Ключові поняття: стандартна бібліотека шаблонів, контейнер, ітератор, алгоритм, адаптер, функціональний об’єкт, послідовні контейнери, асоціативні контейнери, контейнери-адаптери, псевдо контейнери.

Стандартна бібліотека шаблонів (STL) (англ. Standard Template Library) – набір узгоджених узагальнених алгоритмів, контейнерів, засобів доступу до їх вмісту і різних допоміжних функцій в C ++.

У бібліотеці виділяють п'ять основних компонентв:

– Контейнер (англ. Container) – зберігання набору об'єктів в пам'яті.

– Ітератор (англ. Iterator) – забезпечення засобів доступу до вмісту контейнера.

– Алгоритм (англ. Algorithm) – визначення обчислювальної процедури.

– Адаптер (англ. Adaptor) – адаптація компонентів для забезпечення різного інтерфейсу.

– Функціональний об'єкт (англ. Functor) – приховування функції в об'єкті для використання іншими компонентами.

Контейнери бібліотеки STL можна розділити на чотири категорії: послідовні, асоціативні, контейнери-адаптери і псевдоконтейнери.

У таблицях 1 – 4 наведено характеристку контейнерів бібліотеки STL.

Таблиця 1 – Послідовні контейнери

Назва	Опис контейнеру
vector	Динамічний масив довільного доступу з автоматичною зміною розміру при додаванні або видаленні елементу. Доступ до елементів здійснюється за індексом. Підтримує додавання-видалення елементів.
list	Двусвязний список, елементи якого зберігаються в довільних шматках пам'яті, на відміну від контейнера vector, де елементи зберігаються в безперервній області пам'яті. Пошук перебором повільніший, ніж у vector через більший час доступу до елементу. Доступ до елементів здійснюється за індексом.
deque	Контейнер схожий на vector, але з можливістю швидкої вставки і видалення елементів на обох кінцях. Реалізований у вигляді двусвязанного списку лінійних масивів. На відміну від vector, дек не гарантує розташування всіх своїх елементів в безперервній ділянці пам'яті, що робить неможливим безпечне використання арифметики покажчиків для доступу до елементів контейнеру.

Таблиця 2 – Асоціативні контейнери

Назва	Опис контейнеру
set	Впорядкована множина унікальних елементів. Забезпечує стандартні

	операції над множинами типу об'єднання, перетину, віднімання.
multiset	Те ж що і set, але дозволяє зберігати повторювані елементи.
map	Упорядкований асоціативний масив пар елементів, що складаються з ключів та відповідних їм значень. Ключі повинні бути унікальні. Порядок проходження елементів визначається ключами.
multimap	Те ж що і map, але дозволяє зберігати кілька однакових ключів.

Таблиця 3 – Контейнери-адаптери

Назва	Опис контейнеру
stack	Стек - контейнер, в якому додавання і видалення елементів здійснюється з одного кінця.
queue	Черга - контейнер, з одного кінця якого можна додавати елементи, а з іншого - виймати.
priority_queue	Черга з пріоритетом, організована так, що найбільший елемент завжди стоїть на першому місці.

Таблиця 4 – Псевдоконтейнери

Назва	Опис контейнеру
bitset	Служить для зберігання бітових масок. Схожий на vector фіксованого розміру. Розмір фіксується тоді, коли оголошується об'єкт bitset. Ітераторів в bitset немає. Оптимізовано за розміром пам'яті.
basic_string	Контейнер, призначений для зберігання і обробки рядків. Зберігає в пам'яті елементи поспіль єдиним блоком, що дозволяє організувати швидкий доступ до всієї послідовності. Елементи повинні бути простих (фундаментальних) типів даних. Визначено конкатенація за допомогою +.
valarray	Шаблон служить для зберігання числових масивів і оптимізований для досягнення підвищеної обчислювальної продуктивності. В деякій мірі схожий на vector, але в ньому відсутня більшість стандартних операцій.

У контейнерах для зберігання елементів використовується семантика передачі об'єктів за значенням. Іншими словами, при додаванні контейнер отримує копію елементу. Якщо створення копії небажано, то використовують контейнер покажчиків на елементи. Присвоєння елементів реалізується за допомогою оператора присвоювання, а їх руйнування відбувається з використанням деструктора.

Розглянемо приклад використання контейнеру vector:

```
#include <iostream>
#include <vector>
int main()
{

std::vector<int> vect;
for (int count=0; count < 5; ++count)
```

```
vect.push_back(10 - count); // вставка чисел у кінець масиву
for (int index=0; index < vect.size(); ++index)
std::cout << vect[index] << ' ';
std::cout << '\n';
}
```

С початку підключається бібліотека vector, потім оголошується екземпляр vect цілого типу шаблону класу vector. Далі здійснюється доступ до методів push_back(), size() класу vector.

Приклад використання контейнеру deque:

```
#include <iostream>
#include <deque>

int main()
{
std::deque<int> deq;
for (int count=0; count < 4; ++count)
{
deq.push_back(count); // вставка чисел в кінець масиву
deq.push_front(10 - count); // вставка чисел на початок масиву
}
for (int index=0; index < deq.size(); ++index)
std::cout << deq[index] << ' ';
std::cout << '\n';
}
```

? Контрольні запитання:

1. Що таке STL?
2. Які основні компоненти використовуються у бібліотеці STL?
3. Які існують категорії контейнерів бібліотеки STL?
4. Які класи бібліотеки STL належать до послідовних контейнерів?

✍Практичні завдання:

1. Використайте шаблон vector для масиву даних про авто.
2. Використайте шаблон list для двусвязного списку даних про авто.
3. Використайте шаблон deque для обліку даних про черги авто на заправці.
4. Використайте шаблон set для побудови двох множин цілих чисел і обчислення їх перетину.
5. Використайте шаблон multiset для підрахунку числа входжень кожного числа в безліч цілих чисел з повторами.
6. Використайте шаблон map для виключення повторів серед безлічі цілих чисел.

7. Використайте шаблон multimap для виключення повторів комбінацій серед безлічі пар цілих чисел.

8. Використайте шаблон stack для стека дійсних чисел.

9. Використайте шаблон queue для черги авто на мийці.

10. Використайте шаблон priority_queue для черги замовлень, щоб обслуговувати найбільші замовлення в першу чергу.

11. Використайте шаблон bitset для зберігання інформації про простоту перших 10000 цілих чисел.

12. Використайте шаблон basic_string для зберігання прізвищ імен та по батькові.

13. Використайте шаблон valarray для масиву даних про авто.

14. Використайте шаблон hash_map для масиву даних про авто.

15. Використайте шаблон unordered_map для масиву даних про авто.

Тести з теми «Стандартна бібліотека STL: контейнерні класи»

1. Який з перелічених компонент не є стандартним компонентом бібліотеки STL?

А.
контейнер;
Б.
адаптер;
В.
алгоритм;
Г.
функціональний об’єкт;
Д.
файл.

2. До якого типу контейнерів бібліотеки STL належать класи vector, list, deque?

А.
до послідовних;
Б.
до асоціативних;
В.
до контейнерів-адаптерів;
Г.
до псевдо контейнерів.

3. До якого типу контейнерів бібліотеки STL належать класи set, multiset, map, multimap?

А.
до послідовних;
Б.

до асоціативних;

В.

до контейнерів-адаптерів;

Г.

до псевдо контейнерів.

4. До якого типу контейнерів бібліотеки STL належать класи stack, queue, priority_queue?

А.

до послідовних;

Б.

до асоціативних;

В.

до контейнерів-адаптерів;

Г.

до псевдо контейнерів.

5. До якого типу контейнерів бібліотеки STL належать класи bitset, basic_string, valarray?

А.

до послідовних;

Б.

до асоціативних;

В.

до контейнерів-адаптерів;

Г.

до псевдо контейнерів.

Тема 14. Стандартна бібліотека STL: ітератори, шаблонні алгоритми.

Ключові поняття: стандартна бібліотека STL, ітератори.

У бібліотеці STL для доступу до елементів в якості посередника вікористовується узагальнена абстракція, іменована ітератором. Кожен контейнер підтрімує свій вид ітератору.

Ітератор – інтерфейс, який надає доступ до елементів конкретного контейнеру та навігацію за ними.

Існує п'ять категорій ітераторів в залежності від операцій, визначених для них:

– ввод (input iterators);
– виведення (output iterators);
– однонаправлені (forward iterators);
– двонаправлені (bidirectional iterators);
– довільного доступу (random access iterators.).

Ітератор вводу – це такий ітератор, який переміщається тільки вперед і підтримує тільки читання.

Ітератор вводу – найпростіший різновид ітераторів. Такі ітератори доступні тільки для читання. Приклад заповнення контейнера STL з текстового файлу використовуючи ітератор вводу:

```
#include <algorithm> //бібліотека з алгоритмом  for_each
#include <iostream>
 using namespace std;
 void printValue(int num) //цю функцію необхідно передати в алгоритм for_each
{
  cout << num << "\n";
}
 void main(void)
{
  int Arr[] = {1, 2, 3, 4, 5}; //массив
  for_each(Arr, Arr + 5, printValue); }
```

for_each – це алгоритм, який приймає покажчик на початок масиву, покажчик на перший елемент за границею масиву і покажчик на функцію.

Ітератор виведення – це ітератор для посилання на область пам'яті, куди виводяться дані.

Приклад виводу елементів контейнеру на екран:

```
#include <iostream>
#include <vector> //без этого не сработает
```

```
using namespace std;
int main(void)
{
int Arr[] = {1, 2, 3, 4, 5};
copy(Arr,Arr+5,ostream_iterator<int>(cout,"\n"));
return 0;
}
```

Однонаправлений ітератор – поєднання ітераторів вводу/виводу. Цей ітератор може переміщатися по ланцюжку елементів в одному напрямку.

Двонаправлений ітератор – ітератор, який поєднує ітератори вводу/виводу і здатен переміщатися в обох напрямках завдяки інкременту (++) і декременту (-).

Ітератор довільного доступу – ітератор, який дає легко звертатися до довільного елементу**.**

Алгоритми STL

Алгоритми STL реалізовані у вигляді глобальних функцій, які працюють з використанням ітераторів. Для їх роботи потрібно підключити заголовочний файл algorithm.

Алгоритми min_element () і max_element () знаходять мінімальний і максимальний елементи в контейнері:

```
#include <iostream>
#include <list>
#include <algorithm>

int main()
{
std::list<int> li;
for (int nCount=0; nCount < 5; ++nCount)
li.push_back(nCount);
std::list<int>::const_iterator it;
it = min_element(li.begin(), li.end());
std::cout << *it << ' ';
it = max_element(li.begin(), li.end());
std::cout << *it << ' ';}
std::cout << '\n';
}
```

У наступному прикладі використовується алгоритм find (), щоб знайти певне значення в списку, а потім – функція list :: insert () для додавання нового значення в список:

```
#include <iostream>
```

```
#include <list>
#include <algorithm>

int main()
{
std::list<int> li;
for (int nCount=0; nCount < 5; ++nCount)
li.push_back(nCount);

std::list<int>::iterator it;
it = find(li.begin(), li.end(), 2); // пошук значення 2 у списку
li.insert(it, 7); // вставки числа 7 перед числом 2
for (it = li.begin(); it != li.end(); ++it)
std::cout << *it << ' ';
std::cout << '\n';
}
```

Для сортування вектору використовується функція sort():

```
std::vector<int> vect;
vect.push_back(4);
vect.push_back(8);
vect.push_back(-3);
vect.push_back(3);
vect.push_back(-8);
vect.push_back(12);
vect.push_back(5);
std::sort(vect.begin(), vect.end()); // сортировка елементів вектору
std::vector<int>::const_iterator it; for (it = vect.begin(); it != vect.end(); ++it)
std::cout << *it << ' ';
std::cout << '\n';
std::reverse(vect.begin(), vect.end());
for (it = vect.begin(); it != vect.end(); ++it)
std::cout << *it << ' ';
std::cout << '\n';}
```

Окрім наведених вище алгоритмів у STL існують і інші загальні алгоритми. Ознайомитись з нимим можна, відкривши документацію до цієї бібліотеки.

? Контрольні запитання:

1. Яким чином у бібліотеці STL здійснюється доступ до елементів?
2. Що таке ітератор?
3. Які існують категорії ітераторів бібліотеки STL?
4. Чим відрізняються однонаправлені від двонаправлених ітераторів?
5. Яким чином можна підключити бібліотеку алгоритмів STL?

✍Практичні завдання:

У завданнях теми 13 реалізувати ввод та вивід елементів об’єктів, використовуючи літератори вводу та виводу.

Тести з теми «Стандартна бібліотека STL: контейнерні класи»

1. Оберіть найточніше визначення поняттю «літератор бібліотеки STL»

А.

Ітератор – це об'єкт, який здатний перебирати елементи контейнерного класу без необхідності користувачеві знати реалізацію певного контейнерного класу.

Б.

Ітератор – це покажчик, який дозволяє вводити і виводити дані з різними структурами даних.

В.

Ітератор – шаблони класів, які надають доступ до елементів контейнеру.

2. Що не є категорією ітераторів?

А.

ітератор вводу;

Б.

ітератор виведення*;*

В.

керований ітератор;

Г.

двонаправлений ітератор.

3. Ітератор довільного доступу – це…

А.

ітератор, який надає доступ до будь-якого елементу контейнеру у будь-якому направленні;

Б.

ітератор, який надає доступ до будь-якого елементу контейнеру в одному направленні;;

В.

ітератор, який надає доступ до максимального елементу контейнеру у будь-якому направленні;;

Г.

ітератор, який надає доступ до мінімального елементу контейнеру у будь-якому направленні.

4. Який заготовочний файл необхідно підключити, щоб використовувати алгоритми бібліотеки STL?

А.
algorithm;
Б.
algorithms*;*
В.
library;
Г.
stl.

5. Яка функція використовується у STL для впорядкування елементів?

А.
sort();
Б.
esort();
В.
sort_e();

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Stroustrup B. The Design and Evolution of C++. Boston : Addison–Wesley, 1994. 472 с.

2. Грицюк Ю.І., Рак Т.Є. Об'єктно-орієнтоване програмування мовою С++. Львів : ЛДУ БЖД, 2011. 404 с.

3. Бублик В.В. Об’єктно-орієнтоване програмування. Київ : ІТкнига, 2015. 624 с.

4. Грицюк Ю.І., Рак Т. Є. Програмування мовою С++. Львів : ЛБУ БЖД, 2011. 292 с.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

Основна

1. Карпенко Н.В. Розробка програм на мові С у сучасних середовищах. Дніпропетровськ : Ліра, 2016. 144 с.

2. Трофименко О.Г., Прокоп Ю.В., Швайко І.Г. та ін. С++. Теорія та практика. Одеса : ВЦ ОНАЗ, 2011. 587 с.

3. Луцик Ю.А., Комличенко В.Н. Объектно-ориентированное программирование на языке С++. Минск : БГУИР, 2008. 266 с.

Додаткова

1. Медведев В.И. Особенности объектно-ориентированного программирования на С++/CLI, C# и Java. Казань : Школа, 2010. 444 с.

2. Ревотюк М.П. Объектно-ориентированное программирование и проектирование. Минск : БГУИР, 2014. 194 с.

3. Дубов И.Р., Быков В.И. Языки программирования. Объектно-ориентированное программирование. Владимир : Изд-во ВлГУ, 2018. 96 с.

4. Ноткин А.М. Объектно-ориентированное программирование: ООП на языке С++. Пермь : Пермский национальный исследовательский политехнический университет, 2013. 230 с.

ЗМІСТ

Навчальне видання
(українською мовою)

Решевська Катерина Сергіївна
Лісняк Андрій Олександрович
Борю Сергій Юрійович

ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

Навчальний посібник
для здобувачів ступеня вищої освіти бакалавра
спеціальності «Комп’ютерні науки»
освітньо-професійної програми «Комп’ютерні науки»

Рецензент С.Чопоров
Відповідальний за випуск С.Ю. Борю
Коректор К.С. Решевська