Лабораторна робота № 10. Шаблони.

Мета роботи: вивчення шаблонів та їх застосування.

Теоретичний матеріал лекції, [1, розділ 10], [2, розділ 18], [3, розділ 16]

1. Короткі теоретичні відомості

Шаблонна функція – узагальнене визначення функції, за яким компілятор автоматично згенерує код конкретного екземпляру функції. Шаблонна функція оголошується за допомогою кваліфікатора `template`. Параметризований тип задається ключовим словом `class` або `typename`.

```
template <class T1,…,class Tn>
тип_функції імя_функції (T1 t1, … Tn tn);
```

Ключове слово `export` перед оголошенням `template` дозволяє використовувати шаблон з іншого файлу, повторюючи його оголошення а не все визначення.

Ключове слово `typename` може:

- замінити ключове слово `class` в оголошенні шаблону, тобто задавати узагальнений тип;
- інформувати компілятор про те, що деяке ім'я використовується в оголошенні шаблонного класу як ім'я типу, а не об'єкту, наприклад

```
typename X::Name someObject;
```

Тут імя `X::Name` використовується як ім'я типу.

Приклад шаблонної функції для обміну значень двох змінних:

```
template <class T>
void swap(T& t1, T& t2) {
   T t=t1;
   t1=t2;
   t2=t;
}
```

Під час виклику шаблонної функції її формальні параметри типів набувають значень відповідних фактичних аргументів. Якщо фактичний аргумент є об'єктом класу, то, як правило, виникає необхідність у перевантаженні деяких операцій.

Шаблонний клас – це узагальнене визначення множини класів, з якого компілятор автоматично згенерує код конкретного класу. Шаблонний клас реалізує один з варіантів поліморфізму – поліморфізм типів. Оголошення шаблонного класу:

```
template <список параметрів шаблону>
class імя_класу
{
   протокол класу, який використовуэ параметри шаблону
}
```

У списку параметрів шаблону можуть бути *параметризовані і спеціалізовані* типи. Якщо метод шаблонного класу визначається поза протоколом класу, то у заголовку метода вказуються оголошення формальних параметрів, наприклад

```
template <class T, int k>
A<T, k>::~A()
{ delete []p; }
```

Параметри типу можуть набувати стандартних значень або бути типами, визначеними користувачем. Можна визначити параметр-посилання на змінну заданого типу:

```
template <int & ref>
class X {
…
};
```

Параметри шаблону можуть набувати значення за замовчуванням. Якщо один з параметрів

набуває значення за замовчуванням, то усі наступні повинні бути параметрами за замовчуванням:

```
template <class T=int, int k=0>
class A {
   …
public:
   A(int n);
   …
}
```

Тоді при оголошенні об'єкта фактичні аргументи шаблону, які набувають значення за замовчуванням, можна не задавати:

```
void main() {
   A<> a(5);
   A<unsigned> b(10);
   A<short, 7> c(20);
}
```

Допускається вкладеність шаблонів при оголошенні об'єктів параметризованих класів:

```
template <class T>
class X {…};
template <int n>
class Y {…};
int main() { X<Y<5>> x; }
```

Методи шаблонного класу можуть бути шаблонними функціями:

```
template <class T1, class T2>
class X {
public:
   template <class U, class V>
      void mf(const U &u, const V &v) {}
};
```

Визначаючи шаблонний метод поза класом, необхідно повторити `template`-оголошення для кожної групи параметрів типу:

```
template <class T1, cass T2>
template <class U, class V>
   void X<T1, T2>::mf(const U &u, const V &v){}
```

Статичні елементи шаблонних класів є спільними для усіх об'єктів з однаковими значеннями аргументів шаблону.

1.1. Друзі шаблонних класів

Друзями шаблонних класів можуть бути зовнішні функції, методи інших класів, інші класи. Якщо дружня функція використовує об'єкт шаблонного класу, то перед її заголовком необхідно розмістити `template`-оголошення:

```
template <class T>
class A {
   friend void f1();
template <class T>
   friend void f2(A<T>&);
template <class U>
   friend void f3(U&);
};
// звичайна функція, друг множини класів, відношення 1:n
void f1();
// функція-друг одного типу T, відношення 1:1
template <class T>
void f2(A<T>& a) {}
// параметризована дружня функція, відношення m:n
template <class U>
void f3(U& u) {}
```

```
void main() {
   f1();
   A<int> a;
   f2(a);
   double x;
   f3(x);
}
```

Друзями шаблонного класу можуть бути методи інших класів:

```
template <class T> class A;
template <class T>
class D {
public:
template <class U>
   void f3(U& u) {}     // шаблонний метод шаблонного класу
};

template <class T>
class C {
public:
   void f2(A<T>& a) {} // звичайний метод шаблонного класу
};

class B {
public:
   void f1() {}        // звичайний метод нешаблонного класу
};
template <class T>
class A {
   friend void B::f1();
   friend void C::f2(A<T>&);
   template <class T>
   template <class U>
      friend void D<U>::f3(U&);
};
void main() {…}
```

1.2. Перевантаження операцій шаблонних класів

Під час роботи з шаблонними класами виникає необхідність у перевантаженні операцій. Приклад перевантаження бінарної операції `operator+` як члена класу:

```
template <class T>
class A {
   T x;
public:
   A(){}
   A(T x){this->x=x;}
   A<T> operator+(A<T>&);
   void print(){cout<<x<<endl;}
};
template <class T> A<T> A<T>::operator+(A<T>& a) {
   return A<T>(x+a.x);
}
void main() {
   A<int> a(2),b(3),c;
   c=a+b;
   c.print();
}
```

1.3. Віртуальні методи шаблонних класів

Шаблонний клас може містити оголошення віртуальних методів. Шаблонні методи (з оголошенням template) не можуть бути віртуальними.

Віртуальні методи забезпечують пізнє зв'язування, якщо вони викликаються за допомогою вказівників (або посилань) на базовий клас, проініціалізованих адресою (або ідентифікатором об'єкта – для посилань) похідного класу з public успадкуванням. Для шаблонних класів вказівник (або посилання) на базовий клас та об'єкт повинні мати однакову параметризацію.

Приклад програми. Створити шаблонний клас `vector` з двома параметрами: перший є параметром типу, а другий – цілочисловим значенням для ініціалізації елементів вектора. Визначити конструктор, деструктор, методи пошуку екстремальних значень (мінімуму та максимуму), впорядкування, обчислення евклідової норми. Перевантажити операції +, =, [], <<, >> відповідно для додавання, присвоєння, контролю діапазону індексу та введення об'єктів.

```
#include <iostream>
#include <cstdlib>
#include <cstring>
#include <math.h>

using namespace std;

template <class T, int k>
class vector
{
  //T *v;
  int size;
public:
  T *v;
  vector(int newsize);
  ~vector();

  int getsize() { return size; }

  T extr(char *);
  vector& sort(char *);
  double norma(void);

  vector& operator+(vector&);
  T& operator[] (int index);
  vector& operator=(const vector&);

  template <class Y>
    friend ostream& operator<<(ostream&, vector&);
  template <class Y>
    friend istream& operator>>(istream&, vector&);
};


template <class T, int k>
  vector<T,k>::vector(int newsize)
  {
    v=new T[size=newsize];
    for(int i=0;i<size;i++) v[i]=k;
  }


  template <class T, int k>
  vector<T,k>::~vector() { delete []v; }


  template <class T, int k>
  T vector<T,k>::extr(char * MinOrMax)
  {
    T ExtrElem=v[0];
```

```
for(int i=0;i<size;i++)
{
  if (!strncmp(MinOrMax,"min",3))
  {
  if(v[i]<ExtrElem) ExtrElem=v[i];
  }
  else if(v[i]>ExtrElem) ExtrElem=v[i];

}
  return ExtrElem;
}

template <class T, int k>
vector<T,k>& vector<T,k>::sort(char * UpOrDown)
{
  T x;
  for(int i=0;i<size-1;i++)
  for(int j=i+1;j<size;j++)
    if( !strncmp(UpOrDown,"max",3) )
    {
      if(v[i]>v[j]) {
        x=v[i];
        v[i]=v[j];
        v[j]=x;
      }
    }
    else
      if(v[i]<v[j]) {
        x=v[i];
        v[i]=v[j];
        v[j]=x;
      }
      return *this;
}

template <class T, int k>
 double vector<T,k>::norma(void)
 {
  double s=0;
  for(int i=0;i<size;i++) s+=v[i]*v[i];
  return sqrt(s);
  }


template <class T, int k>
vector<T,k>& vector<T,k>::operator+(vector<T,k>& y)
{
  if(size!=y.size) false;
  for(int i=0;i<size;i++) v[i]+=y.v[i];
  return *this;
}

template <class T, int k>
T& vector<T,k>::operator[] (int index)
{
  if(index<0 || index>=size)
  throw "Індекс за межами діапазону можливих значень";
  return v[index];
}

template <class T, int k>
vector<T,k>& vector<T,k>::operator=(const vector<T,k>& x)
{
  if(this!=&x)
```

```
      {
      delete []v;
      v=new T[size=x.size];
      for(int i=0;i<size;i++) v[i]=x.v[i];
      }
      return *this;
    }

    template <class T>
    istream& operator>>(istream& is, vector<T,0>& x)
    {
      cout<<"Введіть "<<x.getsize()<<" елементів вектора\n";
      for(int i=0;i<x.getsize();i++) is>>x.v[i];  // ???
      return is;
    }

    template <class T>
    ostream& operator<<(ostream& os, vector<T,0>& x)
    {
      for(int i=0;i<x.getsize();i++) os<<x[i]<<' ';
      os<<endl;
      return os;
    }

    int main() {
      int n=5;
      vector<int,0> V(n),U(n);
      vector<int,0> *Z=new vector<int,0>(n);

      char s1[]="min", s2[]="max";

      try
      {
      cin>>V;
      cin>>U;
      *Z=V+U;
      cout<<"Сума " << *Z;
      cout<<"Мінімальний елемент " <<Z->extr(s1)<<endl;
      //cout<<"Максимальний елемент "<<Z->extr(s2)<<endl;
      Z->sort(s2);
      cout<<"Сума відсортована "<<*Z;
      cout<<"Норма "<<Z->norma()<<endl;
      }
      catch(bool)
      {
      cout<<"Різна кількість елементів векторів"<< endl;
      }
      catch(char *s)
      {
      cout<<s<<endl;
      }

    }
```

```
Введіть 5 елементів вектора
3 23 65 3 9
Введіть 5 елементів вектора
12 9 34 66 3
Сума 15 32 99 69 12
Мінімальний елемент 12
Сума відсортована 12 15 32 69 99
Норма 126.313
```

Запитання.

1. Що таке шаблонна функція і як вона оголошується?
2. Що таке шаблонний клас і як він оголошується?
3. Що таке параметризовані і спеціалізовані типи в шаблонах функцій і класів?
4. Які можуть бути друзі у шаблонних методів?
5. Перевантаження операцій, як членів шаблонних класів.
6. Ключові слово `typename` і `export`.

Завдання.

1. Написати шаблонний клас для послідовного пошуку елементів МАСИВУ за заданим ключем. Якщо елемент знайдено, то на екран виводиться уся інформація, що відповідає заданому ключу. Застосувати клас для пошуку елементів різних типів.

2. Створити шаблонний клас для роботи з ФАЙЛАМИ даних різних типів. Перевизначити методи відкривання файлу, запису даних у файл, зчитування даних з файлу. Створити файл рядків та файл дійсних чисел. Знайти найдовший рядок та найбільше дійсне число у створених файлах.

3. Написати шаблонний клас для сортування одновимірного МАСИВУ за зростанням значень елементів. Застосувати цей клас для сортування масивів цілих та дійсних чисел, масивів та рядків символів.

4. Побудувати шаблонний клас ВПОРЯДКОВАНИЙ МАСИВ. Включити елементи у масив так, щоб не порушити впорядкованості масиву. Вилучити елемент з масиву. Вивести масив на екран. Застосувати шаблон для роботи з різними типами елементів масивів.

5.Написати шаблонний клас, який знаходить контрольну суму ЕЛЕМЕНТА довільного типу. Контрольна сума – це кількість одиниць у машинному зображенні заданого елемента.

6. Створити параметризований клас однозв’язного СПИСКУ. Тип елемента списку визначається параметром шаблона. Передбачити функції для виконання таких операцій: створення нового елемента списку на його початку; вилучення першого елемента списку; створення нового елемента списку у його кінці; вилучення останнього елемента списку; визначення кількості елементів списку.

7. Створити параметризований однонаправлений лінійний кільцевий СПИСОК. Тип елемента списку визначається параметром шаблону. Передбачити функції для виконання таких операцій: занесення елемента списку у список; вилучення елемента зі списку; виведення елементів списку на екран; визначення кількості елементів списку.

8. Написати шаблонний клас, який створює копію двовимірного динамічного МАСИВУ довільного типу. Передбачити операції копіювання за рядками, за стовпцями, копіювання вибраного рядка або стовпчика, копіювання головної діагоналі. Розробити функції потокового введення-виведення масивів.

9. Створити клас СТЕК, який ґрунтується на статичному масиві вказівників. Тип елемента стеку визначається параметром шаблону. Передбачити функції для виконання таких операцій: занесення елементів у стек; вилучення значення з верхівки стеку; виведення усіх значень стеку на екран; повернення кількості елементів стеку.

10. Розробити шаблонний клас СТЕК. Побудувати два стеки із впорядкованих елементів даних. Виконати злиття стеків в один стек так, щоб не порушити загальної впорядкованості елементів.

11. Створити клас БЛОКНОТ, параметризований структурованими типами даних. Визначити конструктори, деструктор, методи роботи з даними. На основі класу БЛОКНОТ створити клас ЖУРНАЛ обліку продукції на складі. Вивести список товарів та їх кількість.

12. Розробити шаблонний клас КОНТРОЛЕР файлової системи для керування доступом до файлів і каталогів. Використати КОНТРОЛЕР для роботи з об’єктами класів ФАЙЛ і КАТАЛОГ. Виконати пошук, копіювання, перейменування, витирання, виведення вмісту файла

або каталога на екран. Визначити необхідні дані, конструктори, деструктори, методи та перевантажені операції. Передбачити опрацювання можливих помилок.\

13. Розробити клас ПРИВІД оптичного диску, який може бути параметризований об'єктами класів CD або DVD. Забезпечити ідентифікацію об'єктів CD/DVD та виведення їхнього вмісту на еран у вигляді списку файлів та каталогів. Визначити необхідні дані, конструктори, деструктори, методи та перевантажені операції.

2. Приклади для самостійної роботи

```
/* listing 1 - приклад шаблонної функції, яка міняє місцями дві змінні незалежно
від їх типу. Позначення шаблонної функції:
template <class Тип> return_тип імя_функції(список параметрів узаг. типів)
або
template <typename Тип> return_тип імя_функції(список параметрів узаг. типів)
class або typename – синоніми.
*/
#include <iostream>
using namespace std;

// Шаблонна функція
template <class X> void swapargs(X &a, X &b)
//template <typename X> void swapargs(X &a, X &b)

{
  X temp;

  temp = a;
  a = b;
  b = temp;
}

int main()
{
  int i=10, j=20;
  double x=10.1, y=23.3;
  char a='x', b='z';

  cout << "Початкові значення i, j: " << i << ' ' << j << '\n';
  cout << "Початкові значення x, y: " << x << ' ' << y << '\n';
  cout << "Початкові значення a, b: " << a << ' ' << b << '\n';

  swapargs(i, j); // переставлення цілих    integers
  swapargs(x, y); // переставлення дійсних  floats
  swapargs(a, b); // переставлення символів chars

  cout << "Переставлені значення i, j: " << i << ' ' << j << '\n';
  cout << "Переставлені значення x, y: " << x << ' ' << y << '\n';
  cout << "Переставлені значення a, b: " << a << ' ' << b << '\n';

  return 0;
}
Початкові значення i, j: 10 20
Початкові значення x, y: 10.1 23.3
Початкові значення a, b: x z
Переставлені значення i, j: 20 10
Переставлені значення x, y: 23.3 10.1
Переставлені значення a, b: z x

/* listing 5 – шаблонна функція з двома узагальненими типами */
#include <iostream>
using namespace std;
/* шаблонна функція записана в два рядки */
```

```
template <class type1, typename type2>
void myfunc(type1 x, type2 y)
{
  cout << x << ' ' << y << '\n';
}

int main()
{
  myfunc(10, "Я вчу C++");

  myfunc(98.6, 19L);

  return 0;
}
```

```
10 Я вчу C++
98.6 19
```

```
/* listing 6 – явне перевантаження шаблонної функції */
#include <iostream>
using namespace std;

// Шаблонна функція
template <class X> void swapargs(X &a, X &b)
{
  X temp;

  temp = a;
  a = b;
  b = temp;
  cout << "всередині шаблону swapargs.\n";
}

// Заміщення шаблонної функції swapargs() для цілих чисел
void swapargs(int &a, int &b)
{
  int temp;

  temp = a;
  a = b;
  b = temp;
  cout << "Всередині спеціалізації функції swapargs для цілих\n";
}

/* ---------------------------------------------------------------
нова синтаксична конструкція для явного заміщення (спеціалізації) параметрів
шаблонної функції
template<> void swapargs<int>(int &a, int &b)
{
  int temp;

  temp = a;
  a = b;
  b = temp;
  cout << "Всередині спеціалізації функції swapargs для цілих\n";
}
--------------------------------------------------------------- */

int main()
{
  int i=10, j=20;
  double x=10.1, y=23.3;
  char a='x', b='z';

  cout << "Початкові значення i, j: " << i << ' ' << j << '\n';
```

```
  cout << "Початкові значення x, y: " << x << ' ' << y << '\n';
  cout << "Початкові значення a, b: " << a << ' ' << b << '\n';

  swapargs(i, j); // виклик явно перевантаженої функції swapargs()
  swapargs(x, y); // виклик узагальненої функції swapargs()
  swapargs(a, b); // виклик узагальненої функції swapargs()

  cout << "Переставлені значення i, j: " << i << ' ' << j << '\n';
  cout << "Переставлені значення x, y: " << x << ' ' << y << '\n';
  cout << "Переставлені значення a, b: " << a << ' ' << b << '\n';

  return 0;
}
```

Початкові значення i, j: 10 20
Початкові значення x, y: 10.1 23.3
Початкові значення a, b: x z
Всередині спеціалізації функції swapargs для цілих
всередині шаблону swapargs.
всередині шаблону swapargs.
Переставлені значення i, j: 20 10
Переставлені значення x, y: 23.3 10.1
Переставлені значення a, b: z x

/* listing 8 - перевантаження шаблонної функції створенням шаблонів, які відрізняються списком параметрів */

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

// Перша версія шаблонної функції f()
template <class X> void f(X a)
{
  cout << "Всередені функції f(X a)\n";
}

// Друга версія шаблонної функції  f()
template <class X, typename Y> void f(X a, Y b)
{
  cout << "Всередині функції f(X a, Y b)\n";
}

int main()
{
  f(10);     // виклик функції f(X)
  f(10, 20); // виклик функції f(X, Y)

  return 0;
}
```

Всередені функції f(X a)
Всередині функції f(X a, Y b)

/* listing 9 – змішування узагальнених і стандартних параметрів */

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

const int TABWIDTH = 8;

// Виведення на екран даних з позиції tab
template <class X> void tabOut(X data, int tab)
{
  for(; tab; tab--)
    for(int i=0; i<TABWIDTH; i++) cout << ' ';

    cout << data << "\n";
```

```
}

int main()
{
  tabOut("Перевірка", 0);
  tabOut(100, 1);
  tabOut('X', 2);
  tabOut(10/3, 3);

  return 0;
}
```

Перевірка
100
X
3

```
/* обмеження на узагальнені функції. Узагальнені функції повинні мати однакове тіло для всіх реалізацій.
/* listing 10 – наступні функції не можна замінити на узагальнені, так як вони мають різне тіло і різне призначення */
#include <iostream>
#include <cmath>
using namespace std;

void myfunc(int i)
{
  cout << "Значення дорівнює: " << i << "\n";
}

void myfunc(double d)
{
  double intpart;
  double fracpart;

  fracpart = modf(d, &intpart);
  cout << "Дробова частина: " << fracpart;
  cout << "\n";
  cout << "Ціла частина: " << intpart;
}

int main()
{
  myfunc(1);
  myfunc(12.2);

  return 0;
}
```

Значення дорівнює: 1
Дробова частина: 0.2
Ціла частина: 12

```
//  застосування узагальнених (шаблонних) функцій
/* listing 11 - узагальнене сортування методом бульбашки */
#include <iostream>
using namespace std;

template <class X> void bubble(
  X *items,  // вказівник на впорядковуваний масив
  int count) // число елементів в масиві
{
  register int a, b;
  X t;
```

```
  for(a=1; a<count; a++)
    for(b=count-1; b>=a; b--)
      if(items[b-1] > items[b]) {
        // переставлення елементів
        t = items[b-1];
        items[b-1] = items[b];
        items[b] = t;
      }
}

int main()
{
  int iarray[7] = {7, 5, 4, 3, 9, 8, 6};
  double darray[5] = {4.3, 2.5, -0.9, 100.2, 3.0};

  int i;

  cout << "Невпорядкований масив цілих чисел: ";
  for(i=0;  i<7; i++)
    cout << iarray[i] << ' ';
  cout << endl;

  cout << "Невпорядкований масив дійсних чисел double: ";
  for(i=0;  i<5; i++)
    cout << darray[i] << ' ';
  cout << endl;

  bubble(iarray, 7);
  bubble(darray, 5);

  cout << "Відсортований масив цілих чисел: ";
  for(i=0;  i<7; i++)
    cout << iarray[i] << ' ';
  cout << endl;

  cout << "Відсортований масив дійсних чисел double: ";
  for(i=0;  i<5; i++)
    cout << darray[i] << ' ';
  cout << endl;

  return 0;
}
```

```
Невпорядкований масив цілих чисел: 7 5 4 3 9 8 6
Невпорядкований масив дійсних чисел double: 4.3 2.5 -0.9 100.2 3
Відсортований масив цілих чисел: 3 4 5 6 7 8 9
Відсортований масив дійсних чисел double: -0.9 2.5 3 4.3 100.2
```

```
/* listing 12 - узагальнена функція ущільнення масиву  */
#include <iostream>
#include <string.h>

using namespace std;

template <class X> void compact(
  X *items,  // вказівник на ущільнюваний масив
  int count, // число елементів в масиві
  int start, // індекс першого вилученого елементу
  int end)   // індекс останнього вилученого елементу
{
  register int i;

  for(i=end+1; i<count; i++, start++)
    items[start] = items[i];
```

```
  /* з демонстраційною метою незаповнена частина масиву обнуляється */
  for( ; start<count; start++) items[start] = (X) 0;
}

int main()
{
  int nums[] = {0, 1, 2, 3, 4, 5, 6};
  char str[] = "Common function";

  int i,N1,N2;

  cout << "size(int)=" << sizeof(int) << "\n";
  N1=sizeof(nums)/4;
  cout << "Неущільнений int масив size=" << N1 << " :";
  for(i=0;  i<N1; i++)
    cout << nums[i] << ' ';
  cout << endl;

  cout << "size(char)=" << sizeof(char) << "\n";
  N2=strlen(str);
  cout << "Неущільнений символьний рядок size(string)=" << N2 << " :";
  for(i=0;  i<N2; i++)
    cout << str[i] << ' ';
  cout << endl;

  compact(nums, N1, 2, 4);
  compact(str, N2, 6, 10);

  cout << "Ущільнений integer масив: ";
  for(i=0;  i<7; i++)
    cout << nums[i] << ' ';
  cout << endl;

  cout << "Ущільнений символьний рядок: ";
  for(i=0;  i<N2; i++)
    cout << str[i] << ' ';
  cout << endl;

  return 0;
}
```

```
size(int)=4
Неущільнений int масив size=7 :0 1 2 3 4 5 6
size(char)=1
Неущільнений символьний рядок size(string)=15 :C o m m o n   f u n c t i o n
Ущільнений integer масив: 0 1 5 6 0 0 0
Ущільнений символьний рядок: C o m m o n t i o n
```

```
/* Оголошення узагальненого класу
template <class Тип> class імя_класу
{
…
}
Створення обєкту узагальненого класу
імя_класу <типЮ імя_обєкту;
*/

/* listing 13 - узагальнений стек */
#include <iostream>
using namespace std;

const int SIZE = 10;
```

```
// Створення узагальненого класу стек
template <class StackType> class stack {
  StackType stck[SIZE];     // містить елементи стека
  int tos;                  // індекс верхівки стека

public:
  stack() { tos = 0; }      // ініціалізація стека
  void push(StackType ob);  // заштовхування обєкту в стек
  StackType pop();          // виштовхування об'єкту зі стека
};

// реалізація функції push - заштовхування об'єкта в стек
template <class StackType>
void stack<StackType>::push(StackType ob)
{
  if(tos==SIZE) {
    cout << "Стек повний\n";
    return;
  }
  stck[tos] = ob;
  tos++;
}
// реалізація функції pop - виштовхування зі стеку
template <class StackType>
StackType stack<StackType>::pop()
{
  if(tos==0) {
    cout << "Стек порожній\n";
    return 0; // якщо стек порожній, то повертається null
  }
  tos--;
  return stck[tos];
}

int main() {
 // стек символів
 stack <char> s1, s2;
 int i;
 s1.push('a');
 s2.push('x');
 s1.push('b');
 s2.push('y');
 s1.push('c');
 s2.push('z');

 for(i=0; i<3; i++) cout << "Виштовхування s1: " << s1.pop() << "\n";
 for(i=0; i<3; i++) cout << "Виштовхування s2: " << s2.pop() << "\n";

 // стек дійсних чисел
 stack <double> ds1, ds2;
 ds1.push(1.1);
 ds2.push(2.2);
 ds1.push(3.3);
 ds2.push(4.4);
 ds1.push(5.5);
 ds2.push(6.6);

for(i=0; i<3; i++) cout << "Виштовхування ds1: " << ds1.pop() << "\n";
for(i=0; i<3; i++) cout << "Виштовхування ds2: " << ds2.pop() << "\n"

  return 0;
}
```

Виштовхування s1: c

```
Виштовхування  s1: b
Виштовхування  s1: a
Виштовхування  s2: z
Виштовхування  s2: y
Виштовхування  s2: x
Виштовхування  ds1: 5.5
Виштовхування  ds1: 3.3
Виштовхування  ds1: 1.1
Виштовхування  ds2: 6.6
Виштовхування  ds2: 4.4
Виштовхування  ds2: 2.2
```

```
/* listing 18 – використання класом двох узагальнених типів */
#include <iostream>
using namespace std;

template <class Type1, class Type2> class myclass
{
  Type1 i;
  Type2 j;
public:
  myclass(Type1 a, Type2 b) { i = a; j = b; }
  void show() { cout << i << ' ' << j << '\n'; }
};

int main()
{
  myclass<int, double> ob1(10, 0.23);
  myclass<char, char *> ob2('X', "Шаблони потужний механізм");

  ob1.show(); // show int, double
  ob2.show(); // show char, char *

  return 0;
}
```

```
10 0.23
X Шаблони потужний механізм
```

```
/* listing 19 – комбінування перевантаженого оператора [] з шаблонним класом для створення узагальненого безпечного масиву */
#include <iostream>
#include <cstdlib>
using namespace std;

const int SIZE = 10;

template <class AType> class atype {
  AType a[SIZE];
public:
  atype() {
    register int i;
    for(i=0; i<SIZE; i++) a[i] = i;
  }
  AType &operator[](int i);
};

// Перевірка діапазону масива для об'єкту  Atype.
template <class AType> AType &atype<AType>::operator[](int i)
{
  if(i<0 || i> SIZE-1) {
    cout << "\nЗначення індекса ";
    cout << i << " виходить за допустимі межі.\n";
    exit(1);
  }
```

```
  return a[i];
}

int main()
{
  atype<int> intob; // масив integer
  atype<double> doubleob; // масив double

  int i;

  cout << "Масив integer: ";
  for(i=0; i<SIZE; i++) intob[i] = i;
  for(i=0; i<SIZE; i++) cout << intob[i] << "  ";
  cout << '\n';

  cout << "Масив Double: ";
  for(i=0; i<SIZE; i++) doubleob[i] = (double) i/3;
  for(i=0; i<SIZE; i++) cout << doubleob[i] << "  ";
  cout << '\n';

  intob[12] = 100; // генерація повідомлення про помилку під час виконання

  return 0;
}
```

Масив integer: 0 1 2 3 4 5 6 7 8 9
Масив Double: 0 0.333333 0.666667 1 1.33333 1.66667 2 2.33333 2.66667 3
Значення індекса 12 виходить за допустимі межі.

```
/* listing 20 – використання стандартних типів в узагальнених класах.
В шаблонних параметрах можна використовувати стандартні типи */
#include <iostream>
#include <cstdlib>
using namespace std;

// Аргумент int size є стандартним
template <class AType, int size> class atype {
  AType a[size]; // довжина масиву передається через параметр size
public:
  atype() {
    register int i;
    for(i=0; i<size; i++) a[i] = i;
  }
  AType &operator[](int i);
};

// Перевірка діапазону для об'єкту atype.
template <class AType, int size>
AType &atype<AType, size>::operator[](int i)
{
  if(i<0 || i> size-1) {
    cout << "\nЗначення індекса ";
    cout << i << " виходить за допустимі межі\n";
    exit(1);
  }
  return a[i];
}

int main()
{
  atype<int, 10> intob;       // масив integer розміром 10
  atype<double, 15> doubleob; // масив double розміром  15

  int i;
```

```
  cout << "Масив integer: ";
  for(i=0; i<10; i++) intob[i] = i;
  for(i=0; i<10; i++) cout << intob[i] << "  ";
  cout << '\n';

  cout << "Масив double: ";
  for(i=0; i<15; i++) doubleob[i] = (double) i/3;
  for(i=0; i<15; i++) cout << doubleob[i] << "  ";
  cout << '\n';

  intob[12] = 100; // генерація повідомлення про помилку

  return 0;
}
```

```
Масив integer: 0  1  2  3  4  5  6  7  8  9
Масив double: 0  0.333333  0.666667  1  1.33333  1.66667  2  2.33333  2.66667  3
3.33333  3.66667  4  4.33333  4.66667
Значення індекса 12 виходить за допустимі межі
```

```
/* listing 23 – використання аргументів за замовчуванням у шаблонних класах */
#include <iostream>
#include <cstdlib>
using namespace std;

// Параметр типу AType по замовчуванню дорівнює int, а змінна i дорівнює 10.
template <class AType=int, int size=10> class atype {
  AType a[size]; // розмір масиву передається аргументом size
public:
  atype() {
    register int i;
    for(i=0; i<size; i++) a[i] = i;
  }
  AType &operator[](int i);
};

// Перевірка діапазону для об'єкту atype.
template <class AType, int size>
AType &atype<AType, size>::operator[](int i)
{
  if(i<0 || i> size-1) {
    cout << "\nЗначення індекса ";
    cout << i << " виходить за межі\n";
    exit(1);
  }
  return a[i];
}

int main()
{
  atype<int, 5> intarray;  // integer масив, розміром 100
  atype<double> doublearray; // double  масив, розміром size
  atype<> defarray;      // по замовчуванню оголошено int масив розміром 10

  int i;

  cout << "int масив: ";
  for(i=0; i<5; i++) intarray[i] = i;
  for(i=0; i<5; i++) cout << intarray[i] << "  ";
  cout << '\n';

  cout << "double масив: ";
  for(i=0; i<10; i++) doublearray[i] = (double) i/3;
  for(i=0; i<10; i++) cout << doublearray[i] << "  ";
  cout << '\n';
```

```
  cout << "масив по замовчуванню: ";
  for(i=0; i<10; i++) defarray[i] = i;
  for(i=0; i<10; i++) cout << defarray[i] << "  ";
  cout << '\n';

  return 0;
}
int масив: 0  1  2  3  4
double масив: 0  0.333333  0.666667  1  1.33333  1.66667  2  2.33333  2.66667  3
масив по замовчуванню: 0  1  2  3  4  5  6  7  8  9

/* listing 25 -  явна спеціалізація (заміщення типів) узагальненого класу
задається конструкцією template<> */
#include <iostream>
using namespace std;

template <class T> class myclass {
  T x;
public:
  myclass(T a) {
    cout << "Всередині узагальненого класу\n";
    x = a;
  }
  T getx() { return x; }
};

// Явна спеціалізація типу int для узагальненого класу
template <> class myclass<int> {
  int x;
public:
  myclass(int a) {
    cout << "Всередині спеціалізації  myclass<int>\n";
    x = a * a;
  }
  int getx() { return x; }
};

int main()
{
  myclass<double> d(10.1);
  cout << "double: " << d.getx() << "\n\n";

  myclass<int> i(5);
  cout << "int: " << i.getx() << "\n";

  return 0;
}
Всередині узагальненого класу
double: 10.1
Всередині спеціалізації  myclass<int>
int: 25
```