Лабораторна робота №2. Оголошення та структура класу.

Мета роботи: вивчення засобів введення, редагування та компіляції програм, стандартних потоків введення/виведення, вбудованих типів даних, оголошення класу та рівнів захисту.

Теоретичний матеріал лекції, [1, розділ 2], [2, розділ 12], [3, розділ 10].

1. Короткі теоретичні відомості

Організація однофайлової програми С+:

```
# директиви препроцесора включення файлів
Задання простору імен
Оголошення прототипів функцій;
Оголошення глобальних типів, констант та змінних;
/* допоміжні функції */
Тип_функці Імя_функції(Список формальних параметрів)
{
    оголошення локальних типів, констант, змінних;
   інструкції;
   return вираз;
}
…
/* основна функція */
int main()
{
    Оголошення локальних типів, констант та змінних;
    інструкції;
    return 0;
}
```

Включення файлів у програму:

```
#include <iostream>   // потокове введення-виведення
#include <cstdlib>    // стандартна бібліотека
#include <cmath>      // математичні функції
#include <cstring>    // робота із символьними стрічками
```

Задання простору імен `std` або користувача:

```
using namespace std;
using namespace MySpace;
```

Компіляція однофайлової програми С++ з використанням компілятора (ОС Linux) `g++` у пакетному режимі:

1. Запуск програми на мові С++ на компіляцію і компонування виконуваного файлу:

```
g++ prog_1.c -o prog_1
```

2. Запуск на виконання виконуваного файлу:

```
./prog_1 > out
```

3. Прогляд результату виконання:

```
kate out  або  vim out або kwrite out
```

Рекомендована нотація імен для ідентифікаторів програми. Ідентифікатор бажано будувати з декількох об’єднаних слів, кожне з яких починається з великої літери, наприклад `CountEventElements`. Ідентифікатор повинен містити інформацію про призначення змінної і її тип. Інформація про тип змінної записується як префікс.

Таблиця 1. Префікси ідентифікаторів

Префікс	Тип даних
a	масив
ch	символ
i	індекс
l	довге ціле
lp	далекий вказівник
n	ціле
np	ближній вказівник
sz	стрічка, яка закінчується нульовим символом (ASCIIZ-рядок)
w	ціле без знаку довжиною слово
dw	довге ціле без знаку
m	елемент класу

Для організації потокового введення-виведення необхідно виконати включення: `#include <iostream>`. C++ використовує стандартні потокові об'єкти `cin, cout, cerr`.

Для введення з потоку використовується перевантажена операція >> , яка застосовується до потокового об'єкту `cin`:

```
int x;
cin >> x;
```

Для виведення у потік використовується перевантажена операція << , яка прцює з потоковим об'єктом cout:

```
cout << "Виведення рядка" << endl;
```

Для виведення повідомлень про помилки використовується перевантажена операція << , яка працює з потоковим об'єктом `cerr`:

```
cerr << "Помилки виконання" << endl;
```

Інструкція for each, in

Нова інструкція `for each, in` використовується для роботи з колекціями. Колекція - це набір об'єктів одного типу. Колекціями є масиви і рядки символів.

```
int vector[]={-2, 9, 5, -1, 8};
for each ( int x in vector ) cout << x << endl;
```

У межах дії інструкції `for each, in` можна змінити елементи колекції, але не дозволяється доповнення колекції або вилучення її членів.

Типи даних

Прості типи даних: символ `(char)`, цілі `(int, long, long long)`, bool (`true, false`). У булевих виразах відмінне від нуля значення перетворюється на `true`, а нульове - на `false`. У числових виразах, навпаки, `true` перетворюється на 1, а `false` - на 0). Типи з плаваючою крапкою (`float, double`).

Структуровані типи даних: `struct, enum, union, class`.

Посилання - це тип даних, який має властивості розіменованого вказівника:

```
Тип & ідентифікатор1 = ідентифікатор2;
```

Клас - це структурований тип даних, який інкапсулює (обмежує область досяжності) оголошення полів даних та функцій для їх використання. Клас використовується для позначення множини об'єктів, які мають однакову структуру. Структура (зміст) класу називається його *протоколом*. Сукупність методів класу, через які відбувається доступ до елементів класу, називається *інтерфейсом*.

Елементи класу (дані та методи) захищені оголошенням класу:

```
class назва_класу {
public:
   // дані та методи
protected:
```

```
   // дані та методи
private:
   // дані та методи
};
```

Клас C++ визначає три рівні захисту своїх елементів:

- public (відкриті);
- private (закриті) - діє за замовчуванням;
- protected (захищені).

Відкриті елементи досяжні у класі та поза класом у межах дії встановленого простору імен. Закриті елементи можуть використовуватися тільки у межах класу. Захищені елементи доступні у класі та похідних від нього класах.

Клас надає зовнішнім функціях та класам можливість доступу своїх елементів за допомогою методів, розміщених у відкритій частині.

Для ініціалізації даних класу використовується *конструктор* – метод ім'я якого співпадає з іменем класу. Конструктор не повертає значення. Конструктори автоматично викликаються при створенні об'єктів у сегментах коду, стека або у динамічній пам'яті.

У класі за замовчуванням завжди існує конструктор без параметрів (void-конструктор), конструктор копіювання, деструктор та операторна функція присвоєння об'єктів.

Ініціалізація даних конструктором може здійснюватися у списку ініціалізації після символу двокрапка, або в тілі конструктора. Константні поля даних, дані з типом посилання, успадковані дані, дані з типом іншого класу (якщо перекритий конструктор за замовчуванням) ініціалізуються конструктором після символу ':'.

Якщо об'єкт виходить із області досяжності він автоматично знищується методом спеціального призначення – деструктором. Назва деструктора складається з символу "~" та назви класу.

Крім членів класу, оголошення класу може містити прототипи дружніх (friend) функцій або класів. Методи класу та дружні до класу функції мають можливість доступу до елементів усіх частин класу. Рівні захисту діють тільки на членів класу і не діють на друзів класу. Визначати методи та дружні функції можна як у класі, так і поза класом.

Елементи класу (крім друзів, конструкторів, деструктора та операції присвоєння допускають успадкування, яке полягає у використанні оголошень даних та методів базового класу у похідному класі.

У протоколі класу дозволяється оголошувати типи даних за допомогою специфікатора typedef:

```
class Base {
public:
typedef void * PVOID;
Base(int a=0, int b=0, int c=0) {x=a; y=b; z=c;}
~Base(){}
private:
int x,y,x;
};
void main() {
Base::PVOID q;
…
}
```

Запитання.

1. Дати коротке визначення інкапсуляції, поліморфізму і успадкування.
2. Дати характеристику простим типам даних мови C++.
3. Дати характеристику структурованим типам даних мови C++.
4. Для чого використовується анонімне об'єднання.
5. Для чого використовується перелічимий тип.
6. Як працює інструкція for each, in.

7. В чому відмінність між структурою і класом.

8. Для чого використовуються простори імен.

9. Оголошення класу і його синтаксис. Відкриті, закриті і захищені члени класу.

10. Як можна ініціалізувати дані конструктором.

11. Де можна розміщувати функції-члени і як до них звертатися.

12. Як використовуються дружні функції і класи.

Завдання.

1. Написати програму, яка запитує в циклі у користувача ціле число N=10, а потім виводить вказане число зірочками (перший рядок - одна зірочка, другий рядок - дві зірочки і т.д.). При введенні числа більшого від N, програма завершує роботу.

2. Написати програму, яка використовує безіменне об'єднання для переставлення байтів числа `short int` (16 бітів).

3. Написати програму для обчислення виразу $y = a \cdot x^2 + b \cdot x + c$ для комплексних коефіцієнтів *a, b, c* у точці *x*.

4. Припустимо, що є наступне оголошення структури:

```
struct app {
   char name[32];
   int  state[2];
};
```

а) написати функцію, яка приймає структуру `app` як аргумент і відображає її вміст;

б) написати функцію, яка приймає адрес структури `app` як аргумент і відображає її вміст;

в) написати функцію, яка приймає поилання на структуру `app` як аргумент і відображає її вміст.

5. Припустимо, що є наступне оголошення структури:

```
struct app {
char name[32] ;
int  state[2];
};
```

Написати програму, яка розміщує масив з двох структур у динамічній пам'яті, ініціалізує елементи масиву структур значеннями і відображає їх вміст.

6. Припусимо, що функції `f1()` и `f2()` використовують структуру `app` і мають наступні прототипи:

```
void f1(app * a);
const char * f2(const app * a1, const app * a2);
```

Оголосити `p1` як вказівник на функцию `f1`, а `p2` - як вказівник на функцію `f2`. Оголосити `ap` як масив з двох вказівників того ж типу, що і `p1`, і оголосити `pa` як вказівник на масив з трьох вказівників того ж типу, що і `p2`. Використати інструкцію `typedef`. Написати програму, в якій викликаються функції з використанням цих вказівників, а у фунціях відображаються значення отриманих аргументів.

7. Написати програму, яка відображає значення, яке повертає функція `q`

```
double q = calculate (2.5, 10.4, add);
```

яка викликає функцію `add`

```
double add(double x, double y)
{
    return x + y;
}
```

8. Створити клас `box`, конструктору якого передається три значення типу `double`, які є довжинами сторін паралепіпеда. Класс `box` має підраховувати його об'єм і зберігати результат у вигляді значення `double`. Включити в клас метод `vol()`, який буде виводити на екран об'єм любого об'єкта типу `box`.

9. Використовуючи клас

```
#include <ctime>
```

```
class timer {
       clock_t start;
public:
        timer();
       ~timer();
};
```

і стандартну бібліотечну функцію `clock()`, яка повертає число часових тактів з моменту запуску програми створити клас `secmeter` для імітації секундоміра. Якщо поділити число часових тактів на число `CLOCKS_PER_SEC`, то можна отримати значення в секундах. Використати конструктор для початкового встановлення секундоміра в 0. Утворити дві функції-члени `start()` і `stop()` відповідно для запуску і зупинки секундоміру. Включити в класс функцію-член `show()` для виводу на екран величини проміжків часу, які пройшли. Використати деструктор для автоматичного виводу на екран часу, який пройшов з моменту створення об’єкту класу `secmeter`, до його вилучення.

9. Написати програму з використанняс функції, яка знаходить суму своїх аргументів та повертає результат через перший аргумент. Реалізувати варіанти функції, яка дозволяє звернення з різною кількістю аргументів:

- перевантаження функцій;
- функція з параметрами за замовчуванням;
- функція зі змінною кількістю параметрів `f(int &,…)`.

Результати функції повернути через вказівник та через посилання.

10. Написати програму для переписування менших від середнього арифметичного елементів одновимірного масиву дійсних чисел в інший масив. Виділити динамічну пам'ять для збереження масивів. Вхідні дані ввести з клавіатури, результат вивести на екран.

11. В області динамічної пам’яті визначити масив структур з даними про вироби підприємства: дата, код виробу, назва виробу, кількість виробів, ціна одного виробу. Знайти вартість продукції заданого коду, яка обчислюється як сума добутків кількості виробів на ціну одного виробу за всіма входженнями даного коду.

12. Визначити масив рядків символів в області динамічної пам’яті. Кожен рядок символів складається з полів фіксованої довжини: прізвище студента, номер залікової книжки, рейтинг за 100-бальною шкалою. Відсортувати рядки за спаданням рейтингу. Впорядкований масив вивести на екран.

2. Завдання для самостійної робота № 1

```
/* Розд. 12 1. listing 1  - введення-виведення даних типу int у C++*/
#include <iostream>
using namespace std;

int main()
{
  int i;
  cout << "Це вивід.\n";  // коментарій C++
  /* коментарій в стилі C */
  // введення числа за допомогою операції >>
  cout << "Введіть число: ";
  cin >> i;
  // тепер, вивести число за допомогою оператора <<
  cout << i << " в квадраті буде " << i*i << "\n";
  return 0;
}
>Це вивід.
Введіть число: 12
12 в квадраті буде 144

/* 2. listing 10 – введення-виведення даних типу float, double, char*/
#include <iostream>
using namespace std;

int main()
{
  float f;
  char str[80];
  double d;
  cout << "Введіть два числа з плаваючою крапкою: ";
  cin >> f >> d;
  cout << "Введіть символьний рядок: ";
  cin >> str;
  cout << f << " " << d << " " << str;
  return 0;
}
>Введіть два числа з плаваючою крапкою: 123.45 345.123
Введіть символьний рядок: abcdef
123.45 345.123 abcdef

/* 3. listing 13 - оголошенняя str перед використанням */
#include <iostream>
using namespace std;

int main()
{
  float f;
  double d;
  cout << "Введіть два числа з плаваючою крапкою: ";
  cin >> f >> d;
  cout << "Введіть символьний рядок: ";
  char str[80];  // str оголошений перед використанням
  cin >> str;
  cout << f << " " << d << " " << str;
  return 0;
}
>Введіть два числа з плаваючою крапкою: 123.45 345.123
Введіть символьний рядок: abcd efghi
123.45 345.123 abcd

/* 4. listing 27 – клас стек з методами init(), push(), pop() */
```

```
#include <iostream>
using namespace std;
#define SIZE 100

// Створення класу стек
class stack {
  int stck[SIZE];
  int tos;
public:
  void init();
  void push(int i);
  int pop();
};

void stack::init()
{
  tos = 0;
}

void stack::push(int i)
{
  if(tos==SIZE) {
    cout << "Стек повний.\n";
    return;
  }
  stck[tos] = i;
  tos++;
}

int stack::pop()
{
  if(tos==0) {
    cout << "Стек порожній.\n";
    return 0;
  }
  tos--;
  return stck[tos];
}
int main()
{
  stack stack1, stack2;  // створюємо два об’єкти типу stack

  stack1.init();
  stack2.init();

  stack1.push(1);
  stack2.push(55);

  stack1.push(2);
  stack2.push(66);

  cout << stack1.pop() << " ";
  cout << stack1.pop() << " ";
  cout << stack2.pop() << " ";
  cout << stack2.pop() << "\n";

  return 0;
}
```

>**2 1 66 55**

```
/* 5. listing 29 – перевантаження функції аргументами різних типів */
#include <iostream>
using namespace std;
// Функція abs перевантажена три рази
```

```
int abs(int i);
double abs(double d);
long abs(long l);

int main()
{
  cout << abs(-10) << "\n";
  cout << abs(-11.0) << "\n";
  cout << abs(-9L) << "\n";
  return 0;
}
int abs(int i)
{
  cout << "Функція abs() з аргументом типу int\n";
  return i<0 ? -i : i;
}
double abs(double d)
{
  cout << " Функція abs() з аргументом типу double\n";
  return d<0.0 ? -d : d;
}
long abs(long l)
{
  cout << " Функція abs() з аргументом типу long\n";
  return l<0 ? -l : l;
}
>10
Using double abs()
11
Using long abs()
9

/* 6. listing 30 - перевантаження функції аргументами різних типів */
#include <iostream>
#include <cstdio>
#include <cstring>
using namespace std;

void stradd(char *s1, char *s2);
void stradd(char *s1, int i);

int main()
{
  char str[80];
  strcpy(str, "Hello ");
  stradd(str, "there");
  cout << str << "\n";
  stradd(str, 100);
  cout << str << "\n";
  return 0;
}

// зчеплення двох символьних рядків
void stradd(char *s1, char *s2)
{
  strcat(s1, s2);
}
// зчеплення символьного рядка з "stringized" integer
void stradd(char *s1, int i)
{
  char temp[80];
  sprintf(temp, "%d", i);
  strcat(s1, temp);
}
```

```
>Hello there
Hello there100

/* 7. listing 33 – механізм успадкування. Створення класів house і school похідних від класу building */
#include <iostream>
using namespace std;

class building {
  int rooms;
  int floors;
  int area;
public:
  void set_rooms(int num);
  int get_rooms();
  void set_floors(int num);
  int get_floors();
  void set_area(int num);
  int get_area();
};

// клас house є похідним від класу building
class house : public building {
  int bedrooms;
  int baths;
public:
  void set_bedrooms(int num);
  int get_bedrooms();
  void set_baths(int num);
  int get_baths();
};

// клас school є похідним від класу building
class school : public building {
  int classrooms;
  int offices;
public:
  void set_classrooms(int num);
  int get_classrooms();
  void set_offices(int num);
  int get_offices();
};

void building::set_rooms(int num) {
  rooms = num;
}

void building::set_floors(int num) {
  floors = num;
}

void building::set_area(int num) {
  area = num;
}

int building::get_rooms() {
  return rooms;
}

int building::get_floors() {
  return floors;
}

int building::get_area() {
```

```
  return area;
}

void house::set_bedrooms(int num) {
  bedrooms = num;
}

void house::set_baths(int num) {
  baths = num;
}

int house::get_bedrooms() {
  return bedrooms;
}

int house::get_baths() {
  return baths;
}

void school::set_classrooms(int num) {
  classrooms = num;
}

void school::set_offices(int num) {
  offices = num;
}

int school::get_classrooms() {
  return classrooms;
}

int school::get_offices() {
  return offices;
}

int main()
{
  house h;
  school s;

  h.set_rooms(12);
  h.set_floors(3);
  h.set_area(4500);
  h.set_bedrooms(5);
  h.set_baths(3);

  cout << "В будинку " << h.get_bedrooms();
  cout << " спальних кімнат\n";

  s.set_rooms(200);
  s.set_classrooms(180);
  s.set_offices(5);
  s.set_area(25000);

  cout << "В школі " << s.get_classrooms();
  cout << " класних кімнат\n";
  cout << "Її площа дорівнює " << s.get_area();

  return 0;
}
```

>В будинку 5 спальних кімнат
В школі 180 класних кімнат
Її площа дорівнює 25000

```
/* 8. Клас з конструктором і деструктором */
#include <iostream>

using namespace std;
class myclass {
private:
    int a;
public:
      myclass();   // конструтор
      ~myclass();  // деструктор
      void show();
};

myclass:: myclass() {
    cout << ”Вміст конструктора\n”;
    a=10;
}

myclass::~myclass() {
    cout << ”Деструктор\n”;
}

void myclass::show() {
   cout << ”a=” << a << ”\n”;
}

int main() {
    myclass obj;
    ob.show();
    return 0;
}

/* 9. Конструктор і деструтор з таймером */
#incluse <iostream>
#include <ctime>
using namespace std;

class timer {
privet:
    clock_t start;
public:
    timer();
    ~timer();
};

timer::timer()
{
    start=clock();
}

timer::~timer()
{
    clock_t end;
    end=clock();
    cout << ”Затрачений час: ” << (end-start)/CLOCKS_PER_SEC << ”\n”;
}

int main() {
    timer obj;
    char c;
    cout << ”Натисність любу клавішу, а потім Enter: “;
    cin >> c;
    return 0;
}
```

/* 10. Використання об'єднання для побайтного виведення значення типу double у двійковому поданні */

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

union bits {
     bits(double n);
     void show_bits();
     double d;
     unsigned char c[sizeof(double)];
};

bits::bits(double n); {
    d=n;
}

void bits::show_bits() {
    int i, j;
    for(j=sizeof(double)-1; j>=0; j--); {
        cout << "Двійкове подання байту" << j << ":";
        for(i=128;i;i>=1)
            if(i & c)[j]) cout << "1";
            else cout << "0";
        cout << "\n";
    }
}

int main() {
    bits ob(1991.829);
    ob.show();
    return 0;
}
```

/* 11 – анонімне об'єднання */

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

int main() {
    union {
       unsigned char bytes[8];
       double value;
    };

   int I;
   value=859345.324;
   // побайтне виведення тиру double
   for(i=0;i<8;i++) cout << (int) bytes[i] << " ";

   return 0;
}
```

/* 12. listing 37 – клас стек з використання конструкторів і деструкторів */

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

#define SIZE 100

// Створюється клас стек
class stack {
  int stck[SIZE];
  int tos;
public:
  stack();  // constructor
  ~stack(); // destructor
```

```
  void push(int i);
  int pop();
};

// конструктор
stack::stack() {
  tos = 0;
  cout << "Стек ініціалізований\n";
}

// деструктор
stack::~stack() {
  cout << "Стек знищений\n";
}

void stack::push(int i)
{
  if(tos==SIZE) {
    cout << "Стек повний\n";
    return;
  }
  stck[tos] = i;
  tos++;
}

int stack::pop() {
  if(tos==0) {
    cout << "Стек порожній\n";
    return 0;
  }
  tos--;
  return stck[tos];
}

int main()
{
  stack a, b;  // створюємо два об'єкти stack

  a.push(1);
  b.push(55);
  a.push(2);
  b.push(66);

  cout << a.pop() << " ";
  cout << a.pop() << " ";
  cout << b.pop() << " ";
  cout << b.pop() << "\n";

  return 0;
}
```

```
>2 1 66 55
Стек ініціалізований
Стек ініціалізований
2 1 66 55
Стек знищений
Стек знищений
```