Перелік питань до модульних контрольних робіт з дисципліни
“Об’єктно-орієнтоване програмування”

1. Поясніть основні принципи ООП.
2. Поясніть принцип ООП — інкапсуляцію.
3. Визначте поняття класу і об’єкту в ООП.
4. Проаналізуйте поняття відкритих (public) і закритих (private) членів класу в ООП.
5. Визначте загальні форми (синтаксис) описання класу та його функцій-членів.
6. Визначте поняття конструктора класу. Проаналізуйте, коли і як викликаються конструктори?
7. Визначте поняття деструктора класу. Проаналізуйте, коли і як викликаються деструктори?
8. Обгрунтуйте використання конструкторів без та з параметрами.
9. Проаналізуйте, як викликаються конструктори і функції-члени класу.
10. Обгрунтуйте особливості використання вбудованих функцій. Визначте їх переваги та недоліки.
11. Обгрунтуйте особливості використання дружніх функцій.
12. Поясніть принцип ООП — поліморфізм.
13. Проаналізуйте один з видів реалізації статичного поліморфізму — перевантаження функцій.
14. Проаналізуйте один з видів перевантаження функцій — перевантаження конструкторів.
15. Проаналізуйте особливості використання аргументів за замовчуванням.
16. Проаналізуйте особливості перевантаження операторів як функцій-членів класу.
17. Проаналізуйте особливості перевантаження операторів як функцій, дружніх класу.
18. Визначте загальні форми (синтаксис) оператора-функції члена класу та дружньої функції-оператора.
19. Проаналізуйте особливості перевантаження бінарних операторів.
20. Проаналізуйте особливості перевантаження унарних операторів.
21. Проаналізуйте особливості перевантаження операторів відношення та логічних операторів.
22. Поясніть принцип ООП — успадкування.
23. Проаналізуйте особливості доступу до елементів базового класу під час успадкування.
24. Обгрунтуйте використання захищених (protected) членів класу під час успадкування.
25. Поясніть, що відбувається з відкритими (public), захищеними (protected) та закритими (private) членами базового класу, якщо базовий клас успадковується як відкритий (public) похідним?
26. Поясніть, що відбувається з відкритими (public), захищеними (protected) та закритими (private) базового класу, якщо базовий клас успадковується як закритий (private) похідним?
27. Поясніть, що відбувається з відкритими (public), захищеними (protected) та закритими (private) базового класу, якщо базовий клас успадковується як захищений (protected) похідним?
28. Визначте порядок виклику конструкторів і деструкторів при успадкуванні та особливості передачі аргументів конструкторам базового та похідного класів.
29. Визначте особливості виклику конструкторів і деструкторів при множинному успадкуванні.
30. Поясніть множинне успадкування у C++.
31. Проаналізуйте особливості вказівників на базові та похідні класи.
32. Проаналізуйте поняття віртуальної функції як одного із видів реалізації динамічного поліморфізму та визначте поняття поліморфного класу.
33. Проаналізуйте поняття чисто віртуальної функції як одного із видів реалізації динамічного поліморфізму та визначте поняття абстрактного класу.
34. Проаналізуйте відмінності між звичайними віртуальними та чисто віртуальними функціями, між поліморфними і абстрактними класами.
35. Проаналізуйте відмінності між віртуальними та перевантаженими функціями.
36. Обгрунтуйте використання родових функцій.
37. Визначте поняття родової функції та наведіть типову форму(синтаксис) її визначення.
38. Проаналізуйте особливості визначення родової функції кількох родових типів даних.
39. Проаналізуйте відмінності між родовими та перевантаженими функціями?
40. Визначте поняття родового класу. Розкрийте особливості оголошення об’єктів такого класу.
41. Проаналізуйте переваги використання контейнерних класів.
42. Проаналізуйте процес оброблення виняткових ситуацій у мові C++, їх генерацію і перехоплення.
43. Проаналізуйте особливості застосування оператора catch мови C++.
44. Визначте загальні форми (синтаксис) і функції таких операторів мови C++, як: try, catch і throw.
45. Проаналізуйте особливості системи введення-виведення мови C++.
46. Проаналізуйте використання таких засобів форматного введення-виведення у C++, як: прапори формату, функції setf() і unsetf().
47. Проаналізуйте використання таких засобів форматного введення-виведення у C++, як: функції width(), precision(), fill().
48. Визначте особливості використання стандартних маніпуляторів у C++.
49. Визначте особливості оголошення і використання власних маніпуляторів користувача у C++.
50. Визначте особливості оголошення і використання власних функцій вставки у C++.
51. Визначте особливості оголошення і використання власних функцій вилучення у C++.
52. Проаналізуйте особливості файлового введення-виведення у C++.

53. У наведеному коді перевантажте оператор “+” для додавання двох об’єктів класу *coord*:

```
class coord {
   int x,y;
 public:
        coord() { x = 0; y= 0; }
        coord(int i, int j) { x = i; y = j; }
   void get_xy(int &i, int &j) { i = x; j = y; }
  // ... оголошення оператора-функції
};
//... визначення оператора-функції
 void main() { //… оголошення об’єктів
 // … додавання об’єктів - виклик оператора-функції
 }
```

54. У наведеному коді перевантажте оператор “-” для віднімання двох об’єктів класу coord:

```
class coord {
   int x,y;
 public:
        coord() { x = 0; y= 0; }
        coord(int i, int j) { x = i; y = j; }
   void get_xy(int &i, int &j) { i = x; j = y; }
  // ... оголошення оператора-функції
};
//... визначення оператора-функції
 void main() { //… оголошення об’єктів
 // … віднімання об’єктів - виклик оператора-функції
 }
```

55. У наведеному коді перевантажте оператор “*” для множення двох об’єктів класу *coord*:

```
class coord {
   int x,y;
 public:
        coord() { x = 0; y= 0; }
        coord(int i, int j) { x = i; y = j; }
   void get_xy(int &i, int &j) { i = x; j = y; }
  // ... оголошення оператора-функції
};
//... визначення оператора-функції
 void main() { //… оголошення об’єктів
 // … множення об’єктів - виклик оператора-функції
 }
```

56. У наведеному коді перевантажте оператор “/” для ділення двох об’єктів класу *coord*:

```
class coord {
   int x,y;
 public:
        coord() { x = 0; y= 0; }
        coord(int i, int j) { x = i; y = j; }
   void get_xy(int &i, int &j) { i = x; j = y; }
  // ... оголошення оператора-функції
};
//... визначення оператора-функції
 void main() { //… оголошення об’єктів
 // … ділення об’єктів - виклик оператора-функції
 }
```

57. У наведеному коді перевантажте оператор “==” для двох об’єктів класу *coord*:

```
class coord {
   int x,y;
 public:
        coord() { x = 0; y= 0; }
        coord(int i, int j) { x = i; y = j; }
   void get_xy(int &i, int &j) { i = x; j = y; }
  // ... оголошення оператора-функції
};
//... визначення оператора-функції
```

```
 void main() { //… оголошення об’єктів
  // … виклик оператора-функції
}
```

58. У наведеному коді перевантажте оператор “&&” для двох об’єктів класу *coord*:

```
class coord {
   int x,y;
 public:
        coord() { x = 0; y= 0; }
        coord(int i, int j) { x = i; y = j; }
   void get_xy(int &i, int &j) { i = x; j = y; }
  // ... оголошення оператора-функції
};
//... визначення оператора-функції
 void main() { //… оголошення об’єктів
  // … виклик оператора-функції
}
```

59. У наведеному коді перевантажте оператор “++” для об’єкта класу *coord*:

```
class coord {
   int x,y;
 public:
        coord() { x = 0; y= 0; }
        coord(int i, int j) { x = i; y = j; }
   void get_xy(int &i, int &j) { i = x; j = y; }
  // ... оголошення оператора-функції
};
//... визначення оператора-функції
 void main() { //… оголошення об’єктів
  // … інкрементація об’єкта - виклик оператора-функції
}
```

60. У наведеному коді перевантажте оператор “--” для об’єкта класу *coord*:

```
class coord {
   int x,y;
 public:
        coord() { x = 0; y= 0; }
        coord(int i, int j) { x = i; y = j; }
   void get_xy(int &i, int &j) { i = x; j = y; }
  // ... оголошення оператора-функції
};
//... визначення оператора-функції
 void main() { //… оголошення об’єктів
  // … декрементація об’єкта - виклик оператора-функції
}
```

61. У наведеному коді перевантажте оператор “+” як дружню функцію для додавання об’єкта класу *coord* і цілого числа (coord + int):

```
class coord {
   int x,y;
 public:
        coord() { x = 0; y= 0; }
        coord(int i, int j) { x = i; y = j; }
   void get_xy(int &i, int &j) { i = x; j = y; }
  // ... оголошення оператора-функції
};
//... визначення оператора-функції для coord + int
 void main() { //… оголошення об’єктів
  // … додавання об’єкта і числа - виклик
                //оператора-функції
}
```

62. У наведеному коді перевантажте оператор “+” як дружню функцію для додавання цілого числа і об’єкта класу *coord* (int + coord):

```
class coord {
```

```
   int x,y;
  public:
        coord() { x = 0; y= 0; }
        coord(int i, int j) { x = i; y = j; }
    void get_xy(int &i, int &j) { i = x; j = y; }
   // ... оголошення оператора-функції
};
//... визначення оператора-функції для int + coord
  void main() { //… оголошення об'єктів
  // … додавання числа і об'єкта - виклик
                  //оператора-функції
}
```

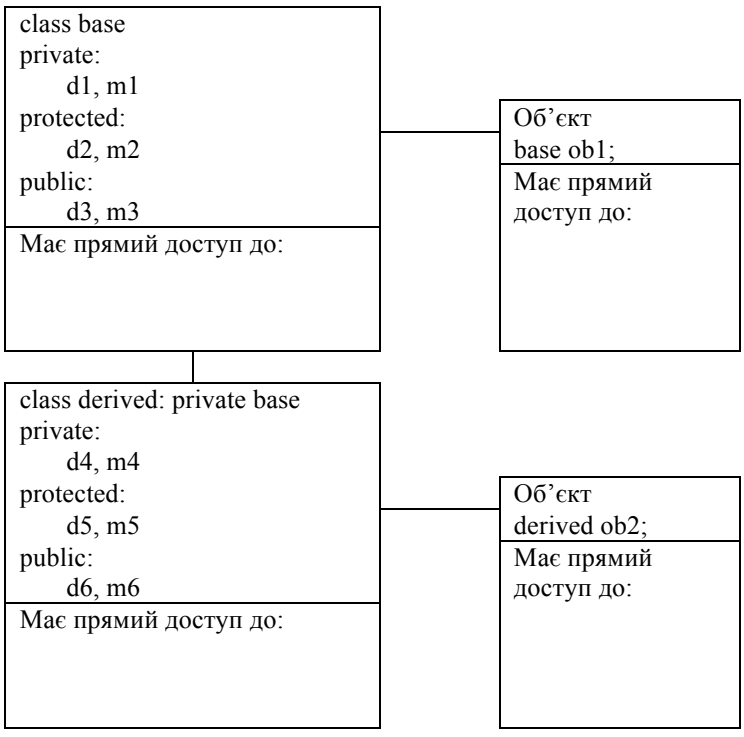
63. У наведеному коді перевантажте оператор вилучення ">>" для об'єкту класу *coord*:

```
class coord {
   int x,y;
  public:
        coord() { x = 0; y= 0; }
        coord(int i, int j) { x = i; y = j; }
    void get_xy(int &i, int &j) { i = x; j = y; }
   // ... оголошення оператора-функції вилучення
};
//... визначення оператора-функції вилучення
  void main() { //… оголошення об'єктів
  // … виклик оператора-функції вилучення
  }
```

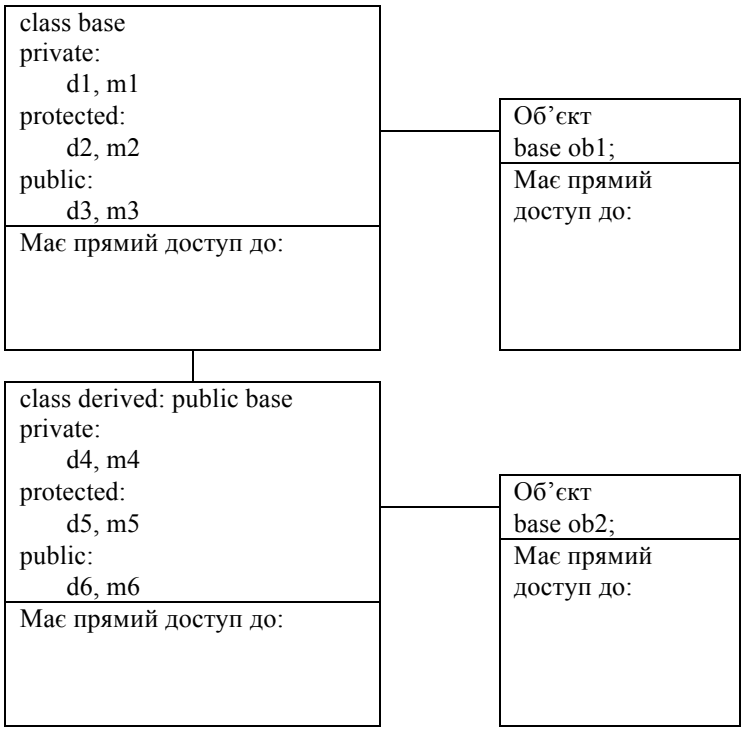
64. У наведеному коді перевантажте оператор вставки "<<" для об'єкту класу coord:

```
class coord {
   int x,y;
  public:
        coord() { x = 0; y= 0; }
        coord(int i, int j) { x = i; y = j; }
    void get_xy(int &i, int &j) { i = x; j = y; }
   // ... оголошення оператора-функції вставки
};
//... визначення оператора-функції вилучення
  void main() { //… оголошення об'єктів
  // … виклик оператора-функції вставки
  }
```

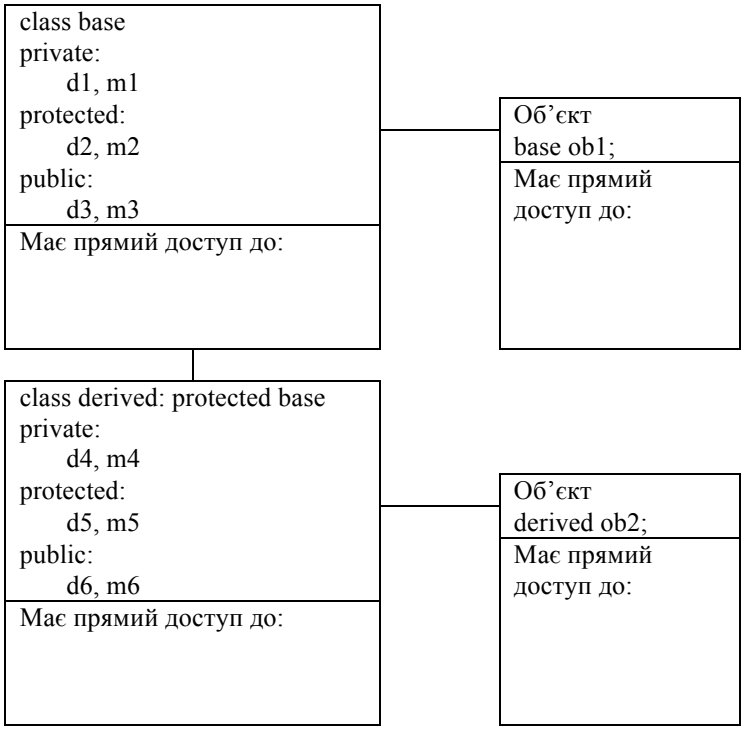
65. Заповніть пропущені ідентифікатори, вказавши у коментарях пояснення:



66. Заповніть пропущені ідентифікатори, вказавши у коментарях пояснення:



67. Заповніть пропущені ідентифікатори, вказавши у коментарях пояснення:



68. Використовуючи принципи потокової системи введення-виведення інформації у C++, напишіть програму в якій: 1) створіть вихідний файл для запису; 2) запишіть до файлу число 100 в десятковій та шістнадцятковій системах числення; 3) закрийте файл; 4) знову відкрийте, але вже як вхідний файл для зчитування; 5) зчитайте інформацію у змінні відповідних типів даних і виведіть її на екран.

69. Напишіть програму у якій: 1) відкрийте вхідний текстовий файл для зчитування; 2) створіть вихідний текстовий файл для запису; 3) скиньте прапор *skipws* вхідного файлового потоку; 4) скопіюйте текстовий файл зчитування у файл запису і при цьому перетворіть пробіли на символи "|"; використайте функцію *eof()* для контролю кінця вхідного файлу зчитування.

70. Напишіть програму, яка виводить на екран таку інформацію у відповідному форматі:

```
          П р и в і т
* * * * П р и в і т
П р и в і т * * * *
1 2 3 . 2 3 4 5 6 7
1 2 3 . 2 3 4 * * *
```

71. Напишіть у поданому нижче коді віртуальну функцію *func()*, яка: 1) у базовому класі base виводить значення змінної і на екран; 2) у похідному класі *derived1* виводить значення квадрату змінної *i* базового класу; 3) у похідному класі *derived2* не перевизначається. Напишіть результат виконання програми:

```
#include<iostream.h>
class base {
  public:
    int i;
    base(int x) {i=x;}
    // додайте визначення віртуальної функції
};
class derived1 : public base {
  public:
    derived1(int x) : base(x) { }
    // додайте визначення віртуальної функції
};
class derived2 : public base {
  public:
    derived2(int x) : base(x) { }
    // у derived2 функція func() не перевизначається
};
void main(){
  base *p;
  base ob(10);
  derived1 d_ob1(10);
  derived2 d_ob2(10);
  p=&ob;
  p->func( );
  p=&d_ob1;
  p->func( );
  p = &d_ob2;
  p->func( );
}
```

72. Напишіть у поданому нижче коді віртуальну функцію *func()*, яка: 1) у базовому класі *base* виводить значення змінної *i* на екран; 2) у похідному класі *derived1* виводить значення квадрату змінної *i* базового класу; 3) у похідному класі *derived2* виводить подвійне значення змінної *i*. Напишіть результат виконання програми:

```
#include <iostream.h>
class base {
  public:
    int i;
    base(int x) {i=x;}
    // додайте визначення віртуальної функції
};
class derived1 : public base {
  public:
    derived1(int x) : base(x) { }
          // додайте визначення віртуальної функції
};
class derived2 : public base {
  public:
    derived2(int x) : base(x) { }
    // додайте визначення віртуальної функції
};
void main()
{
  base *p;
  base ob(10);
  derived1 d_ob1(10);
  derived2 d_ob2(10);
  p=&ob;
  p->func();
```

```
 p=&d_ob1;
 p->func();
 p=&d_ob2;
 p->func();
  }
```

73. Напишіть родову функцію, яка міняє місцями значення двох своїх аргументів. Продемонструйте виклик створеної функції для типів *int*, *char* та *double* у функції *main()*.

74. Напишіть родову функцію з двома родовими типами даних, яка виводить значення двох своїх аргументів на екран. Продемонструйте виклик створеної функції у функції *main()*.

75. Визначте вірне (вірні) твердження:

а) Визначення класів резервує місце в пам'яті; вони створюють нові типи даних, які використовуються у подальшому для оголошення змінних;

б) Функції-елементи, визначені з використанням операції розширення області бачення поза класом, мають область дії цей клас;

в) Функції-елементи, визначені у визначенні класу автоматично вбудовуються. Компілятор зберігає право не вбудовувати будь-яку функцію.

76. Знайдіть помилку (або помилки) в кожному з наступних пунктів і поясніть, як її (їх) виправити:

а) У класі Time є два конструктори

```
Time(int h=0, int m=0, int s=0);
Time( );
```

б)

```
#include <iostream.h>
class count {
friend void setx (count &, int);
public:
count ( ) {x=0;}
void print( ) {cout <<x<<endl;}
protected :
int x;
};
void setx(cout&c, int val) {c.x = val;}
```

в)

```
class base
{   private:
 int x;
    public:
      void print( ){cout <<x<<y;}
    private:
      int y;
};
```

77. Перевантажте бінарний оператор „+" для класу coord так, щоб в результаті додавання отримати добутки відповідних елементів двох об'єктів.

78. Що виводить на екран наступна програма

```
#include <iostream.h>
class A
{public:
      A(){cout<<"Робота конструктора A\n";}
      ~A(){cout<<"Робота деструктора A\n";}
};
class B
{public:
      B(){cout<<"Робота конструктора B\n";}
      ~B(){cout<<"Робота деструктора B\n";}
};
class C:public A, public B
{public:
      C(){cout<<"Робота конструктора C\n";}
      ~C(){cout<<"Робота деструктора C\n";}
};
main()
{C ob;
return 0;
}
```

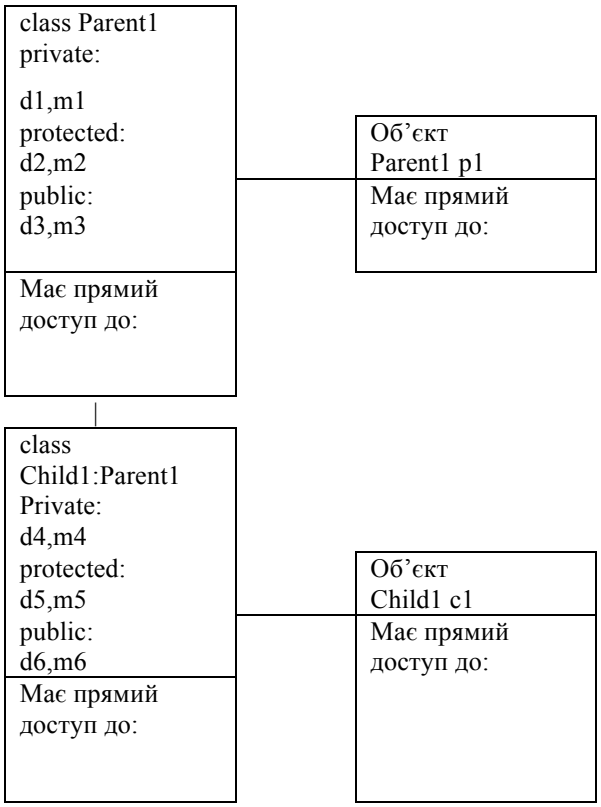
79. Дослідіть наступну конструкцію

```
#include <iostream.h>
class mybase
{ int a,b;
```

```
public:
  int c;
  void setab(int i, int j){a=i; b=j;}
  void getab(int &i, int &j){i=a; j=b;}
};
class derived1:private mybase
{…//тіло класу derived1
};
class derived2:private mybase
{…//тіло класу derived2
};
main()
{ derived1 o1;
  derived2 o2;
   int i,j;
…//тіло main()
}
```

Який із наступних операторів вірний всередині main()
А) o1.getab(i,j);
Б) o2.getab(i,j);
В) o1.c=10;
Г) o2.c=10;

80. Заповніть пропущені ідентифікатори:



81. Визначте вірне (вірні) твердження
а) Специфікатори доступу до елемента завжди закінчуються двокрапкою (:) і можуть зявлятися у визначені класу не однократно і у будь-якій послідовності;
б) Якщо функція-елемент визначена поза визначенням класу, перед ім'ям функції стоїть ім'я класу і бінарна операція розширення області бачення (::);
в) Всі елементи класу після заголовку класу і до першого специфікатора доступу вважаються відкритими.

82. Знайдіть помилку (або помилки) в кожному з наступних пунктів і посніть як її (їх) виправити
а) Допустимо, що в класі Time оголошено наступний прототип
Void ~Time(int);
б) Наступний фрагмент э частиною визначення классу Time

```
class Time
{public:
    …//прототипи функцій
private:
    int hour=0;
    int minute=0;
    int second=0;
};
```

в) Допустимо, що в класі Employee оголошено наступний прототип

```
int Employee(const char*,const char*);
```

83. Перевантажте бінарний оператор „– ” для класу coord так, щоб в результаті віднімання отримати частку від ділення відповідних елементів об’єктів.

84. Що виводить на екран наступна програма

```
#include <iostream.h>
class B1
{public:
    B1(){cout<<”Робота конструктора B1\n”;}
    ~B1(){cout<<”Робота деструктора  B1\n”;}
};
class B2
{public:
    B2(){cout<<”Робота конструктора B2\n”;}
    ~B2(){cout<<”Робота деструктора B2\n”;}
};
class D: public B1, public B2
{public:
    D(){cout<<”Робота конструктора D\n”;}
    ~D(){cout<<”Робота деструктора D\n”;}
};
main()
{D ob;
return 0;
}
```

85. Дослідіть наступну конструкцію

```
#include <iostream.h>
class mybase
{ protected
int a,b;
  public:
int c;
void setab(int i, int j){a=i; b=j;}
void getab(int &i, int &j){i=a; j=b;}
};
class derived1:private mybase
{…//тіло класу derived1
};
class derived2:private mybase
{…//тіло класу derived2
};
main()
{ derived1 o1;
  derived2 o2;
   int i,j;
…//тіло main()
}
```

Який із наступних операторів вірний всередині main()

А) o1.getab(i,j);

Б) o2.getab(i,j);

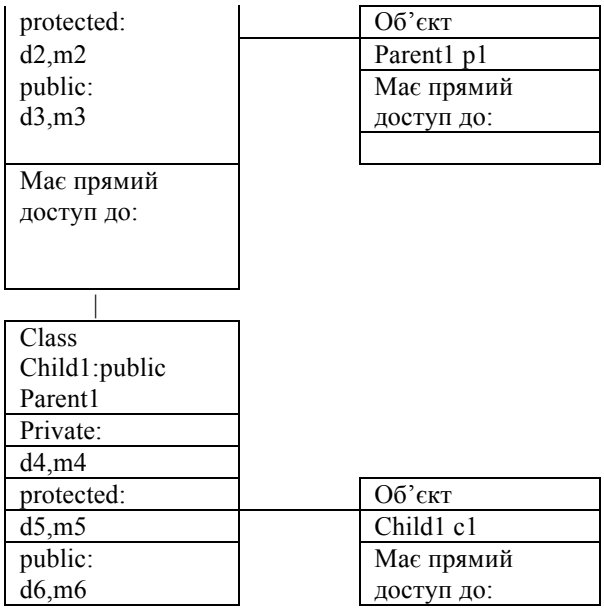
В) o1.c=10;

Г) o2.c=10;

86. Заповніть пропущені ідентифікатори:

```
Class Parent1
private:

d1,m1
```



87. Що виводить на екран наступна програма

```
#include <iostream.h>
class A
{public:
      A(){cout<<"Робота конструктора A\n";}
      ~A(){cout<<"Робота деструктора A\n";}
};
class B
{public:
      B(){cout<<"Робота конструктора B\n";}
      ~B(){cout<<"Робота деструктора B\n";}
};
class C:public A, public B
{public:
      C(){cout<<"Робота конструктора C\n";}
      ~C(){cout<<"Робота деструктора C\n";}
};
main()
{C ob;
return 0;
}
```

88. Дослідіть наступну конструкцію

```
#include <iostream.h>
class mybase
{ int a,b;
public:
  int c;
  void setab(int i, int j){a=i; b=j;}
  void getab(int &i, int &j){i=a; j=b;}
};
class derived1:private mybase
{…//тіло класу derived1
};
class derived2:private mybase
{…//тіло класу derived2
};
main()
{ derived1 o1;
  derived2 o2;
   int i,j;
…//тіло main()
}
```

Який із наступних операторів вірний всередині main()

A) o1.getab(i,j);

Б) o2.getab(i,j);
В) o1.c=10;
Г) o2.c=10;
89. Поясніть, навіщо потрібна категорія захищеності protected?

90. При успадкуванні трьох класів одним напряму, як і коли викликаються конструктори класів?
13. Використовуючи наступну ієрархію класів, створіть конструктор класу C так, щоб він ініціалізував к і передавав аргументи для A() і B().

```
#include <iostream.h>
class A
{int i;
 public:
      A(int a){i=a;}
 }
class B
{int j;
 public:
      B(int a){j=a;}
 }
class C: public A, public B
{int k;
 public:
      /*Створіть C() так, щоб він ініціалізував к і передавав аргументи
в A() і B()*/
      };
```

91. Поясніть, навіщо потрібні віртуальні базові класи?

92. Що виводить на екран наступна програма

```
#include <iostream.h>
class B1
{public:
      B1(){cout<<"Робота конструктора B1\n";}
      ~B1(){cout<<"Робота деструктора  B1\n";}
};
class B2
{public:
      B2(){cout<<"Робота конструктора B2\n";}
      ~B2(){cout<<"Робота деструктора B2\n";}
};
class D: public B1, public B2
{public:
      D(){cout<<"Робота конструктора D\n";}
      ~D(){cout<<"Робота деструктора D\n";}
};
main()
{D ob;
return 0;
}
```

93. Дослідіть наступну конструкцію

```
#include <iostream.h>
class mybase
{ protected
int a,b;
  public:
int c;
void setab(int i, int j){a=i; b=j;}
void getab(int &i, int &j){i=a; j=b;}
};
class derived1:private mybase
{…//тіло класу derived1
};
class derived2:private mybase
{…//тіло класу derived2
```

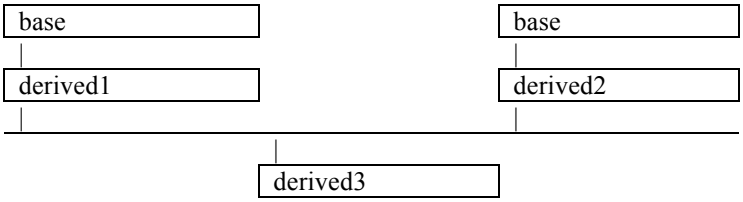
```
};
main()
{ derived1 o1;
  derived2 o2;
   int i,j;
…//тіло main()
}
```

Який із наступних операторів вірний всередині main()
А) o1.getab(i,j);
Б) o2.getab(i,j);
В) o1.c=10;
Г) o2.c=10;

94. Що відбувається, коли захищений член классу успадковується як відкритий?
95. Що відбувається, коли закритий член классу успадковується як захищений?
96. При успадкуванні трьох класів одним класом напряму, як і коли викликаються деструктори класів?
97. Додайте відсутні конструктори в наведену програму

```
#include<iostream.h>
   class base
       {int i,j;
       public:
       //...потрібен конструктор
       void showij(){cout<<i<<' '<<j<<'\n';
       }
       class derived:public base
       {int k;
       public:
       //...потрібен конструктор
       void show(){cout<<k<<' ';
                   showij();}
       }
       int main()
       {
       derived ob(1,2,3);
       ob.show();
       return 0;
       }
```

98. У Вас є конструкція успадкованості класів



Яким чином позбутися неоднозначності в derived3?

99. Що виводить на екран наступна програма

```
#include <iostream.h>
class A
{public:
      A(){cout<<"Робота конструктора A\n";}
      ~A(){cout<<"Робота деструктора A\n";}
};
class B
{public:
      B(){cout<<"Робота конструктора B\n";}
      ~B(){cout<<"Робота деструктора B\n";}
};
class C:public A, public B
{public:
      C(){cout<<"Робота конструктора C\n";}
      ~C(){cout<<"Робота деструктора C\n";}
};
main()
```

```
{C ob;
return 0;
}
```
100. Дослідіть наступну конструкцію
```
#include <iostream.h>
class mybase
{ int a,b;
public:
   int c;
   void setab(int i, int j){a=i; b=j;}
   void getab(int &i, int &j){i=a; j=b;}
};
class derived1:private mybase
{…//тіло класу derived1
};
class derived2:private mybase
{…//тіло класу derived2
};
main()
{ derived1 o1;
   derived2 o2;
    int i,j;
…//тіло main()
}
```
Який із наступних операторів вірний всередині main()
А) o1.getab(i,j);
Б) o2.getab(i,j);
В) o1.c=10;
Г) o2.c=10;

101. Поясніть, навіщо потрібна категорія захищеності protected?

102. При успадкуванні трьох класів одним напряму, як і коли викликаються конструктори класів?

103. Використовуючи наступну ієрархію класів, створіть конструктор класу C так, щоб він ініціалізував к і передавав аргументи для A() і B().
```
#include <iostream.h>
class A
{int i;
  public:
        A(int a){i=a;}
 }
class B
{int j;
  public:
        B(int a){j=a;}
 }
class C: public A, public B
{
   int k;
 public:
  /*Створіть C() так, щоб він ініціалізував і передавав аргументи
   в A() і B()*/
 };
```
104. Поясніть, навіщо потрібні віртуальні базові класи?

105. Чи стане така функція вбудованою? Чому? :
```
inline int function(int x) {
int f=1;
if(x==0||x==1) {  return f;  }
for(int i=x; i>1; i--) {  f*=i;  }
return f;
}
```
106. Що є конструктором(ми) класу *class Gift{ int a,b,c;//...};*
   а) Gift();
   б) void Go();
   в) void Set(int x, int y, int z);

г) Gift(int j, int k);

107. Як компілятор обирає, яку версію перевантаженої функції викликати?

108. Перетворіть функцію *fnazva()* на дружню. (Допишіть все, що потрібно замість „//..” та “/*...*/” - в т.ч. у функції і у її параметрах).

```
class Job{
    char * site;
            int num;
            public:
   //...
};
void nazva( /*  ...   */ ) {
cout << ”site = ” << //... виведіть значення змінної site класу
        cout << ” num = ” << //... виведіть значення змінної num класу
}
```

109. Покажіть усі можливі виклики функції.

```
int v(int a, int b, int c = 4, int d = 3, int f = 2) ;
```

110. Перевантажте конструктор класу так, щоб одна його версія не ініціалізувала змінні класу, інша – ініціалізувала їх за допомогою своїх аргументів (у тому числі продемонструйте використання оператора new). Покажіть виклик перевантаженого конструктора.

```
class A{
  float x, y[10];
  char *z;
public: //...
};
main( ){
//...
}
```

111. Перевантажте дружню функцію до двох класів, одна версія якої повертає суму цілих змінних з обох класів, інша – виводить на екран елемент-рядок та зменшує значення тих самих цілих змінних на число, яке передане функції в якості параметра. Вважайте, що усі необхідні конструктори задано і їх задавати не потрібно. Продемонструйте використання функції (тобто покажіть обидва варіанти її виклику)

```
class MNK{
   int m, n;
   char *k;
public: //...
};
class XYZ{
    int x,y,z;
public: //...
};
```

112. У якому з пунктів(ах) оголошує(ю)ться об’єкт(и) класу *Mark*?

а) Mark a;
б) int Object;
в) Mark a[10];
г) ob Mark;

113. Використовуючи таке визначення конструктора класу *Symvol*, створіть масив із 10 об’єктів, ініціалізуйте змінну *sym* значеннями від ‘A’ до ‘J’.

```
Symvol(char ch) {sym=ch;}
```

114. Як зробити функцію дружньою до двох класів?

115. Використовуючи оператор *new*, покажіть, як динамічно виділити пам’ять під змінну типу *double* і присвоїти їй значення “*-25,01”*. Покажіть також знищення цієї змінної за допомогою оператора *delete*.

116. Яку версію перевантаженої функції обере компілятор під час її виклику (використання) в основній частині програми? В якості відповіді запишіть прототип версії функції.

```
char ftem(char h) { return h; }
int    ftem()  { return 5; }
void ftem(int a, char ch)  { cout << a / ftem() << ” “ << ftem(ch) << ‘\n’; }
main(){
       ftem(50, ‘C‘);
}
```

117. Перевантажте конструктор класу так, щоб одна його версія не ініціалізувала змінні, а інша отримувала б в якості аргументів масив типу *double* та змінну *char*, й ініціалізувала б відповідні змінні класу, а тій змінній, що залишилась, присвоїла б добуток усіх елементів заданого масиву. Покажіть, як викликати перевантажений конструктор.

```
class ABC{
  double a[9];
  double b;
```

```
  char c;
public: //...
};
main() { //...
}
```
118. Перевантажте дружню до двох класів функцію, одна версія якої повертає 0 (нуль), якщо закриті змінні відповідних типів даних двох класів рівні між собою, і 1, якщо навпаки, інша – виводить на екран значення усіх змінних обох класів та збільшує відповідні значення цілих змінних класів у кількість разів, передану версії функції у якості параметра. Продемонструйте використання (виклик) обох версій функції. Вважайте, що усі необхідні конструктори задано і їх визначати не потрібно.
```
class XYZ{
  char x;
  int y;
  char *z;
public: //...
};
class MNK{
  char x;
  int y;
  char *z;
public: //...
};
```
119. Яка функція(ії) використовує(ють) параметр-об’єкт класу Job?
1) Job a(int x);
2) Job f();
3) void f(int x, Job y);
4) Job f(Job object);

120. Чи правильним є такий запис? Поясніть, чому. Якщо не правильно, покажіть, як, на Вашу думку, правильно.
```
inline double price (int a, double b = 5.0, double c = 3.0, double d)  {  return (a * b * c – d);  }
main()
{  cout << price(2.0, 4.0);  }
```
121. Як зробити функцію дружньою до трьох класів?

122. Нехай дано конструктори. Покажіть, як викликати кожний з них.
```
Building() { id = N++;
        adres = new char [255];
        adres = ” “;
        price = 0.0;
}
Building(int _id, double _price) { id = _id;
        adres = new char [255];
        adres = ” “;
        price = _price;
}
Building(char * _adres, double _price)  { id = N++;
                  adres = new char [255];
                  adres = _adres;
                  price = _price;
}
```
123. Яка з версій перевантаженої функції НЕ буде викликана з основної частини програми? У якості відповіді наведіть прототип цієї функції.
```
void fsq(int a) {cout << a * a;}
void fsq(char * ch) { cout << ch << “ “;}
int   fsq(int a, int b) { return a * a – b * b; }
main() { int a = 50, b = 10;
  char *c = “Square = ”;
  fsq(c);
  fsq(b);
}
```
124. Перевантажте конструктор класу так, щоб одна його версія не ініціалізувала змінні класу, інша – ініціалізувала їх за допомогою своїх аргументів (у тому числі продемонструйте використання оператора *new*). Покажіть виклик перевантаженого конструктора.
```
class C {  double x[10];  char z;  int k;
public: //...
};
main(){ //...
```

}
125. Перевантажте дружню функцію до двох класів, одна версія якої повертає добуток відповідних числових змінних обох класів, інша – змінює значення елементів-рядків цих класів на значення, передані їй у якості параметрів. Вважайте, що усі необхідні конструктори задано і їх задавати не потрібно. Продемонструйте використання усіх версій функції

```
class A { double dm, dn; char * ck;
public: //...
};
class B { double dx, dy; char *cz;
public: //...
};
```

126. Оголосіть і визначте повністю конструктор (фактичні дані для закритих змінних передаються конструктору з основної програми) і деструктор для такого класу

```
class Photo {
double x, y;
public:
//...
};
```

127. Яка(і) із функцій використовує(ють) параметр-посилання?

1) int y();
2) void y(int k, int &z);
3) void setx(int *x);
4) int &f();

128. Яким чином дружня функція отримує доступ до закритих змінних класу? (відповідь: „через ключове слово friend" – вважатиметься неповною)

129. Напишіть у позначених коментарях (//...), яка з версій перевантаженої функції *fname()* буде викликана? У коментарях напишіть прототип потрібного варіанту функції

```
void fname(double m) {cout << m;}
void fname(double m, double n) {cout << m * n + m / n << " with: ";}
void fname(char *m) { cout << m << "= ";}
main(){
  double a = 6.0, b = 12.0, c = 10.0;
  char *str = new char[255];
  str = "Name Function ";
  fname(str);   // ...
  fname(b, a); // ...
  fname(c);     // ...
  delete [] str;
}
```

130. Замініть такі перевантажені функції однією з аргументами за умовчанням. Ця одна функція має виконувати те саме, що й дані обидві функції.

```
void show(int a) { cout << a; }
void show(int b, int c) { cout << b + c; }
```

131. Перевантажте конструктор класу так, щоб одна його версія не ініціалізувала змінні, а інша –отримувала б в якості параметрів масив типу *int* та змінну *char* й ініціалізувала б відповідні змінні класу, а тій змінній, що залишилась, присвоїла б добуток усіх елементів заданого масиву. Покажіть, як викликати перевантажений конструктор.

```
class B {
int x[9]; int y;  char z;
public: //...
};
main(){ //...
}
```

132. Перевантажте дружню функцію до двох класів, одна версія якої повертає 1, якщо відповідні закриті змінні рівні між собою і 0, якщо навпаки, інша – збільшує числові елементи на те число, яке передається функції як параметр. Продемонструйте її використання. Вважайте, що усі необхідні конструктори задано.

```
class M{ double d1; char str1; char *ch1;
public: //...
};
class N{ double d1; char str2; char *ch2;
public: //...
};
```

133. Як оголосити родову функцію з трьома родовими типами даних? Наведіть приклад.

134. Що таке віртуальна функція?

135. Перевантажте бінарний оператор „–” для класу coord так, щоб в результаті віднімання (об’єктів) отримати добуток відповідних елементів об’єктів.
136. Виведіть число „20.015” у файл з вирівнюванням по лівому краю, з плюсом, між знаком і числом має бути пробіл. Число повинне займати 8 позицій і відображати 4 знаки після крапки. Символ заповнення ‘*’. Використовуйте функції і прапори класу ios::..
137. Дослідіть наступну конструкцію. Перерахуйте (там, де коментарі //...) усі члени базового і/або похідного класів, до яких мають доступ: 1) функції базового і 2) похідного класів, а також 3)об’єкти базового і 4) похідного класів.

```
class  A { int a;
protected: char * str_a;
public: char ch_a;
void func() { //...1)
      }
};
class B : public A { char * str_b;
protected: int b;
public: char ch_b;
void show() { //...2)
            }
};
main(){ A ob1; //...3)
  B ob2; //...4)
     }
```

138. Що буде виведено на екран?

```
class A{ protected: char * str_a;
public: A(char * _str_a="A") {str_a = new char; strcpy(str_a,_str_a);}
~A(){delete str_a;}
virtual int f(int i = 12) = 0;
};
class B: public A{ public:   int f(int i){ return i+i; }
                        };
class C: public B{ public:
                        };
void main()
{
A * p;
B ob1;
p = &ob1;
cout << p->f();
C ob;
p = &ob;
cout << p->f(5);
}
```

139. Що буде виведено на екран?

```
#include<iostream.h>
#include<conio.h>
class A{
protected:
char *str_a;
public:
A(){ cout << "A()\n"; }
~A(){ cout << "~A()\n"; }
};
class B : protected A{
protected:
char * str_b;
public:
B(){ cout << "B() \n";}
~B(){ cout << "~B() \n"; }
};
class C: protected B{
public:
C(){ cout << "C(): \n"; }
~C(){ cout << "~C(): Bye! \n";  }
};
void main()
```

```
{
cout << "Hello!\n";
C ob_c;
cout << “Bye-bye! \n”;
}
```
140. Що таке родова функція? Приклад.
141. Як передаються параметри конструктору базового класу від похідного?
142. Перевантажте бінарний оператор „+” для класу coord так, щоб в результаті додавання отримати ділення відповідних елементів об’єктів.
143. Дослідіть наступну конструкцію. Перерахуйте (там, де коментарі //...) усі члени базового і/або похідного класів, до яких мають доступ: 1) функції базового і 2) похідного класів, а також 3)об’єкти базового і 4) похідного класів.

```
class  A { protected: int a;
public: char * str_a;
     char ch_a;
void show1() {//....1)
                              }
};
class B : private A { protected: char * str_b;
public: char ch_b;
int b;
void show2() {//....2)
}
};
main(){ A ob1; //...3)
       B ob2; //...4)
}
```

144. Що буде виведено на екран?

```
class A{
protected:
char * str_a;
public:
A(char * _str_a="First") {str_a = new char; strcpy(str_a,_str_a);}
~A(){delete str_a;}
virtual void f() {cout << str_a <<endl;}
};
class B: public A{
public:
     void f() {cout << "Next\n"; }
};
class C: public B{
     };
void main()
{
A *p = new A;
p->f();
B ob1;
p = &ob1;
p->f();
C ob;
p = &ob;
p->f();
}
```

145. Що буде виведено на екран?

```
class A{ protected: char *str_a;
public:     A(){ cout << "A() \n";}
            ~A(){ cout << "~A() \n"; }
};
class B { protected: char * str_b;
public:   B(){ cout << "B() \n"; }
            ~B() { cout << "~B() \n"; }
};
class C: protected B, public A{
public: C(){  cout << "C()\n";    }
~C(){ cout << "~C()\n"; }
```

```
};
void main() { C ob_c; cout << “End! \n”;}
```
146. Які особливості використання вказівників на об’єкти базових класів?
147. Як компілятор визначає, яким реальним типом даних замінити родовий при виклику родової функції?
148. Перевантажте унарний оператор „--” для класу coord так, щоб в результаті декрементування отримати об’єкт, значення відповідних елементів якого отримані за модулем відносно початкових.
149. Дослідіть наступну конструкцію. Перерахуйте (там, де коментарі //...) усі члени базового і/або похідного класів, до яких мають доступ: 1) функції базового і 2) похідного класів, а також 3)об’єкти базового і 4) похідного класів.
```
class  A { int a;
protected: char * str_a;
public:char ch_a;
void show() { //...1)
      }
};
class B : protected A { char * str_b;
protected: int b;
public: char ch_b;
void func() { //... 2)
              }
};
main(){
A ob1; //...3)
B ob2; //...4)
}
```
150. Що буде виведено на екран?
```
class A{protected:char * str_a;
public:
A(char * _str_a="A") {str_a = new char; strcpy(str_a,_str_a); }
~A(){delete str_a;}
virtual int f(int i) = 0;
};
class B: public A{public:
    int f(int i){ return i*i; }       };
class C: public A{public:
    int f(int  i) {return i;}
     };
void main(){
A * p;
B ob1;
p = &ob1;
cout << p->f(5);
C ob;
p = &ob;
cout << p->f(5);
}
```
151. Що буде виведено на екран?
```
class B {protected: char * str_b;
public:   B(){cout << "B(): \n"; }
      ~B() { cout << "~B(): \n"; }
};
class A{ protected:          char *str_a;
public:   A(){cout << "A(): \n";}
      ~A(){ cout << "~A(): \n"; }
};
class C: protected A, public B{ public:
C(){  cout << "C(): \n";  }
~C(){ cout << "~C(): \n"; }
};
void main()
{
cout << "Hello!\n";
C ob_c;
cout << “The End. \n”;
}
```
152. Що таке чисто віртуальна функція? Чи успадковується віртуальна функція?

153. Що таке функція-шаблон? Як її створити і викликати? Наведіть приклад.
154. Перевантажте унарний оператор „++” для класу coord так, щоб в результаті інкрементування отримати добуток від ділення відповідних елементів об’єктів.
155. Дослідіть наступну конструкцію. Перерахуйте (там, де коментарі //...) усі члени базового і/або похідного класів, до яких мають доступ: 1) функції базового і 2) похідного класів, а також 3) об’єкти базового і 4) похідного класів.

```
class  A { int a;
protected: char * str_a;
public: char ch_a;
void show1() {//....1)
}
};
class B : private A { char * str_b;
protected: int b;
public: char ch_b;
void show2() {//....2)
}
};
main(){A ob1; //...3)
B ob2; //...4)
}
```

156. Що буде виведено на екран?

```
class A{protected:char * str_a;
public: A(char * _str_a="A") {str_a = new char; strcpy(str_a,_str_a); }
~A(){delete str_a;}
virtual void f() {cout << str_a <<endl;}
};
class B: public A{public:
          };
class C: public B{public:
    void f() {cout << "Camera \n"  << endl;}
     };
void main()
{A *p = new A("One");
p->f();
B ob1;
p = &ob1;
p->f();
C ob;
p = &ob;
p->f();
}
```

157. Що буде виведено на екран?

```
class B{ protected:           char *str_a;
public:   B(){cout << "B()\n";        }
       ~B(){ cout << "~B()\n"; }
};

class A : protected B{protected:     char * str_b;
public:   A(){cout << "A()\n ";       }
       ~A() { cout << "~A()\n"; }
};
class C: protected A{
public:
C(){cout << "C() \n”;}
~C(){ cout << "~C()\n”; }
};
void main()
{ C ob;
cout<< “Bye!\n”;}
```