

1. Заповніть пробіли для оголошення функції з двома цілими аргументами та повернення їх суми.

```
____ sum(int a, int b)
{
    _____ a _ b;
}
```

2. Заповніть пробіли, щоб оголосити дві цілі змінні " x " та " y ", і переведіть їх у створену раніше функцію " sum () ". Вивести результат на екран.

```
int main()
{
    ____ x = 7;
    int _ = 11;
    cout << ____ (x, y) << endl;
}
```

3. Введіть відсутні ключове слово, щоб оголосити загальний тип даних під назвою "MyType".

```
_____ <class MyType>
```

4. Оголосіть функцію шаблону з двома аргументами. Функція повертає суму своїх аргументів. Аргументи шаблону типу T.

```
_____ <class T>
T sum(_____, a, T b)
{
    _____ a + b;
}
```

```
return T class template public this
```

5. Заповніть пробіли, щоб оголосити дві змінні типу double, і передайте їх функції шаблону "sum ". Вивести результат на екран.

```
int main()
{
    _____ a = 4.3;
    _____ b = 7.2;
    cout << _____ (a, b) << endl;
}
```

6. Заповніть пробіли, щоб оголосити список параметрів шаблону функції функції. Специфікація шаблону має два параметри, названі " T " та " U ".

```
_____ <class T, _____ U>
```

7. Заповніть пробіли, щоб оголосити функцію шаблону ' smaller', взявши два аргументи і повернувши менший. Аргументи мають типи шаблонів " F " і " S " відповідно.

```
_____ <class F, _____ S>
F smaller(F a, S b)
{
    _____ (a < b ? a : b);
}
```

8. Заповніть пробіли, щоб оголосити змінну int та змінну double та передати їх функції шаблону "smaller'. Вивести повернене значення на екран.

```
int main()
{
    int a = 12;
    _____ b = 23.7;
    cout << _____ (a, b) << endl;
}
```

9. Яке твердження про шаблони відповідає дійсності?

- Шаблони є захищеними членами
- Шаблони дозволяють нам оголошувати загальні типи даних
- Шаблони дозволяють нам писати програму без коду

10. Заповніть пробіли, щоб оголосити клас шаблонів "MyClass".

```
_____ <_____ T>
_____ MyClass
{
};
```

11. Оголосіть клас шаблонів "MyClass" 'з двома членами даних шаблону: memOne та memTwo. Ініціалізуйте їх у конструкторі.

```
_____ <class T>
class _____
{
    _____ memOne;
    T _____;
public:
    MyClass(T a, T b) {
        memOne = a;
        memTwo = b;
    }
};
```

```
public T template class memTwo MyClass
```

12. Визначте функцію "MyClass" член-функцію "hello()", де "MyClass" - клас шаблонів, а "hello ()" друкує "Hi" на екрані.

```
_____ <class T>
void MyClass _____::hello()
{
    _____ << "Hi" << endl;
}
```

```
cout template <T> class this
```

13. Заповніть пробіли, щоб визначити функцію "bigger":

```
_____ < _____ T>
T MyClass<T>::bigger()
{
    return (first > second _ first : second);
}
```

14. Заповніть пробіл, щоб оголосити об'єкт " MyClass " об'єктом " integer " як його параметр шаблону.

```
MyClass< ____ > obj;
```

15. Оголосіть шаблон класу "Spunky".

```
_____ <class T>
_____ Spunky {
public:
    Spunky(_____ x) {
        cout << x <<
            " is not a character" << endl; }
};
```

int   template   class   public   protected   T

16. Заповніть пробіли, щоб спеціалізувати клас шаблонів "Spunky" для символів.

```
template ____
class Spunky< _____ >
{
public:
    Spunky(char x) {
        cout << x << "is a char" << endl; }
};
```

17. Заповніть пробіли, щоб оголосити три об'єкти типу "Spunky": i, d, ch, де параметр шаблону i є int, параметр шаблону d - double, а ch - char.

```
int main()
{
    Spunky< __ > i(4);
    Spunky< __ double> d(3.14);
    Spunky< ____ > ch('z');
}
```

18. Оголосіть функцію шаблону з двома аргументами шаблону типу myT. Функція повертає множення своїх аргументів.

```
_____ <class myT>
myT mult( _____ a, myT b)
{
    _____ a * b;
}
```

close myT return class template

19. Заповніть пробіли, щоб оголосити клас шаблонів " MyClass " з двома членами даних шаблону: 'first' та "second". Ініціалізуйте їх у конструкторі.

```
_____ <class T>
_____ MyClass
{
    T first;
    T _____ ;
public:
    MyClass(T a, T b) {
        first = a;
        second = b;
    }
};
```

20. Визначте функцію "MyClass " член-функцію" myFunc () " .

```
_____ <class T>
void MyClass _____ ::myFunc()
{
    _____ << "I love C++" << endl;
}
```

template return class cout <T>

21 Заповніть пробіли, щоб створити клас шаблонів "MyClass" для цілих чисел.

```
template ____  
class MyClass< ____ >  
{  
public:  
    MyClass(int x) {  
        cout << x << endl;  
    }  
};
```