Теоретичен изпит по ООП 2023/2024 учебна година

Въпрос 1: Обяснете основните парадигми на обектно-ориентираното програмиране. За всяка от тях посочете:

- какъв проблем решава;
- какво целим с нейното прилагане;
- как се реализира в C++.

Въпрос 2: Посочете вярното твърдение за дадения по-долу код:

```
class Bar
{
   int var = 0;
public:
   Bar& operator=(const Bar& other)
   {
      var = other.var;
      return *this;
   }
};

void foo(Bar obj)
{
   obj = obj;
}

int main()
{
   Bar obj;
   foo(obj);
}
```

а) В `Bar::operator=` има проблем, компилаторът ще даде грешка, защото не правим проверка за `this != &other`.

б) В `Bar::operator=` има грешка. Не можем да достъпим `other.var`, защото тази променлива е `private` за обекта `other`.

в) Кодът ще се компилира без грешки.

г) В `main` има грешка. Не можем да създадем обекта `obj`, защото в `Bar` не сме дефинирали `default` конструктор, а такъв няма да се генерира автоматично, защото в класа има дефиниран `operator=`.

д) В `main` има грешка. Не можем да подадем `obj` на `foo`, защото в класа няма копиращ конструктор.

Въпрос 3: Даден е следния шаблон на функция:

```
template <typename T>
T sum(const T& a, const T& b)
{ return a + b; }
```

Кои от следните твърдения са верни за него:

а) Може да се направи обръщение по следния начин `sum(3, (int)2.5);`
б) Може да се направи обръщение по следния начин `sum(3, 5);`
в) Може да се направи обръщение по следния начин `sum((int)3, 2.5);`
г) Може да се направи обръщение по следния начин `sum<int>(3, 5);`
д) Може да се направи обръщение по следния начин `sum(int)(3, 5);`
е) Може да се направи обръщение по следния начин `sum(3, 2.5);`
ж) Може да се направи обръщение по следния начин `sum<int>(3, 5.2);`

Въпрос 4: Сравнете динамичния и статичния полиморфизъм в ООП.
За всеки от тях посочете какви проблеми решава и какви слабости има.
За всеки от тях дайте пример, когато е за предпочитане пред другия.

Въпрос 5: Какъв е размерът на една инстанция на дадената по-долу структура?.

```
struct Test
{
   std::uint8_t a;
   std::uint32_t b;

   struct
   {
      std::uint32_t c;
      std::uint16_t d;
   };
};
```

Въпрос 6: Разгледайте следния код. Посочете какви проблеми откривате в него. Дайте предложения за корекции. **Обосновете ги прецизно**.

```
#include <iostream>
#include <assert.h>
using namespace std;
const int max_size = 100;

template <class Type>
class Array final
{
    Type data[max_size];
    int n = 0;

public:
    Array() = default;
```

```
    virtual ~Array() { n = 0; }

    void print()
    {
        for (int i = 0; i < n; ++i)
            cout << data[i] << ' ';
    }

    size_t size() { return n; }

    Array(const Array<Type>& other)
    {
        if (this != &other)
        {
            n = other.n;
            for (int i = 0; i < n; i++)
                data[i] = other.data[i];
        }
    }

    Array& operator=(const Array<Type>& other)
    {
        return (*this = Array(other));
    }

    Array& operator=(const Array<Type>&& other)
    {
        *this = std::move(other);
        return  *this;
    }

    explicit Array(Type data[], int size) : Array()
    {
        assert(size <= max_size);
        n = size;
        for (int i = 0; i < size; ++i)
            this->data[i] = data[i];
    }

    Type operator[](int pos)
    {
        if (pos < n) return data[pos];
        return Type();
    }

    void insert(Type d)
    {
        if (n < max_size)
        {
            data[n++] = d;
        }
    }
};
```

Въпрос 7: Кое от следните е коректният начин да укажем, че функцията `foo` приема константна референция (`const` reference) към обект от тип `Bar`?

а) `void foo(const Bar& ref)`
б) `void foo(Bar const& ref)`
в) И двете са коректни.

Въпрос 8: Посочете вярното твърдение за дадения по-долу код.

```
#include <iostream>
class Test
{
   int& ref;
public:
   Test(int& ref) : ref(ref)
   {}

   void increment() const
   {
      ++ref;
   }
};

int main()
{
   int var = 10;
   Test obj(var);
   obj.increment();
   std::cout << var << "\n";
}
```

а) Ще се получи грешка при изпълнение на програмата (runtime error) -- инкрементираме ref в const функцията increment.
б) Ще се получи грешка при компилиране (compiler error) -- инкрементираме ref в const функцията increment.
в) Кодът ще се компилира без грешки. Програмата извежда 10.
г) Кодът ще се компилира без грешки. Програмата извежда 11.

Въпрос 9: Обяснете предефинирането на оператори в C++. Обърнете внимание на следните неща:

- Кои оператори можем да предефинираме?
- Кои оператори не можем да предефинираме?
- Какви са категориите оператори, които предефинираме?
- Какви са добрите практики при предефиниране на оператори?
- Кога предефинирането на оператори е полезно в ООП?

Въпрос 10: Нека са дадени следните дефиниции:

```
class Test
{
};

void by_ref(Test& param) {}
void by_cref(const Test& param) {}
void by_value(Test param) {}
void by_cvalue(const Test param) {}

int main()
{
   by_ref(Test());
   by_cref(Test());
   by_value(Test());
   by_cvalue(Test());
}
```

Във функцията `main` се съдържат поредица от обръщенията към функциите, при които им подаваме обект от тип `Test`. За всяко от тях посочете дали е коректно или ще предизвика грешка.

```
by_ref(Test());
by_cref(Test());
by_value(Test());
by_cvalue(Test());
```

Въпрос 11: Възможно ли е в един клас да се дефинират няколко различни копиращи конструктора?

Въпрос 12: Каква версия на оператора за присвояване ще генерира компилаторът за дадения по-долу фрагмент?

```
class Test
{
   int var = 0;
};

int main()
{
   Test a, b;
   a = b;
   return 0;
}
```

a) Компилаторът няма да генерира оператор за присвояване, защото променливата `var` е `private`.
b) `Test& operator=(Test);`
c) `Test& operator=(Test &);`

```
   d) Test& operator=(const Test &);
```

Въпрос 13: Какво са нетипови шаблонни параметри?
Какви приложения имат?
Какви проблеми пораждат?
Дайте пример.

Въпрос 14: Какво ще изведе на екрана следната програма?

```
#include <iostream>

class test
{
public:
   void operator++(int)
   {
      std::cout << "A";
   }

   void operator++()
   {
      std::cout << "B";
   }
};

int main()
{
   test obj;
   ++obj;
   obj++;
}
```

Въпрос 15: Какъв вид копиране извършва автоматично генерираният оператор за присвояване в дадения по-долу фрагмент?

```
class Internal
{
   int a;
   Internal& operator=(const Internal& other)
   {
      a = other.a;
      return *this;
   }
};

class External
{
   Internal obj;
   int b;
};

int main()
```

```
{
   External x, y;
   x = y;
}
```

а) Различно, в зависимост от версията на компилатора (compiler specific).
б) Компилаторът не може да генерира оператор за присвояване.
в) Задължително е да се използва покомпонентно копиране с оператора за присвояване на всеки от членовете.
г) Компилаторът може да използва тривиално копиране (например с memcpy или memmove).