ООП. Контрольные вопросы и задания

Макаров П. А.

2023 – 2024 учебный год

1. Найдите и изучите информацию о HTML, CSS, Perl, PHP, Java, JavaScript, SQL, Bash, TEX, Asymptote, Gnuplot. Является ли всё перечисленное языками программирования? Почему? Классифицируйте из данного списка всё, что является языками программирования.

2. Что такое язык разметки? Язык сценариев? Компьютерный язык? Как соотносятся все эти понятия с термином язык программирования?

3. Действительно ли являются компьютерными программами тексты, приведённые в листингах 1 и 2?

```cpp
#include <iostream>

using namespace std;

int main() {
    cout << "Hello, world!" << endl;
    return 0;
}
```

Листинг 1: Пример текста, написанного на языке C++

```python
#!/usr/bin/python3.8

print("Hello, world!")
```

Листинг 2: Пример текста, написанного на языке Python

4. Что такое полнота по Тьюрингу и какое это имеет отношение к программированию?

5. Найдите информацию о наиболее популярных трансляторах языка C++ и изучите их основные особенности. Существуют ли интерпретаторы языка C++?

6. Какие интерпретаторы Python вам известны? К какому типу они относятся и каков принцип их работы?

7. Что такое REPL? Приведите примеры.

8. С какими архитектурами компьютеров вы работали? Каковы их основные особенности? Наидите и изучите информацию о неизвестных вам архитектурах.

9. Назовите существующие интерфейсы взаимодействия пользователя с операционной системой. Каковы их преимущества и недостатки?

10. Определите значения следующих понятий: клиент-серверная архитектура; сетевая модель OSI; стек протоколов TCP/IP; сетевой интерфейс; порт; сокет.

11. Каковы, на ваш взгляд, преимущества и недостатки использования IDE для разработки ПО? С какими инструментами разработчика у вас есть опыт работы?

12. Изучите основные идеи философии UNIX, сформулированные Дугом Макилроем, Майком Ганцарзом и Эриком Рэймондом.

13. Ознакомьтесь с работой Парадигмы программирования Роберта Флойда.

14. В чём разница между параллельным программированием и программированием с разделяемыми данными? Действительно ли это различные концепции? Почему?

15. Какие языки позволяют следовать парадигмам, перечисленным ниже:

    - Императивное программирование.
    - Процедурное программирование.
    - Функциональное программирование.
    - Логическое программирование.
    - ООП.

16. Есть ли существенная разница между императивной и процедурной парадигмами? Какие чисто императивные языки вам известны?

17. Определите значения всех понятий, перечисленных ниже:

- объект, свойства (атрибуты) и методы,
- события,
- интерфейс,
- класс,
- инкапсуляция (encapsulation),
- абстракция данных (data abstraction),
- полиморфизм (polymorphism),
- наследование (inheritance).

18. Изучите листинги 3 и 4. В чём принципиальное различие между ними? Можно ли его устранить? Что для этого потребуется?

```cpp
#include <iostream>

// Polymorphic print function
void print(char * s) {
  std::cout << s << std::endl;
}
void print(char *a[], int N) {
  int i = 0;
  while( i < N ) {
    print(*a++);
    i++;
  }
}

int main(int argc, char **argv) {    // char *argv[] <--- ?
  // Passing a String
  char string[] = "Hello, world!";
  print(string);

  // Passing an Array of Strings
  print(++argv, --argc);    // print(argv++, argc--) <--- ?

  return 0;
}
```

Листинг 3: Пример статического полиморфизма в C++

```python
import sys

var = 'Hello, world!'
print(var)

var = sys.argv
```

```
7 print(var[1:])
```

Листинг 4: Пример полиморфизма в Python

19. Какие парадигмы позволяет использовать программисту современный C++? Python?

20. Изучите самостоятельно механизм декорирования имён (name mangling) в C++. В чём его необходимость и как он связан с перегрузкой имён функций? Какие при этом потенциальные проблемы могут возникнуть при трансляции модульных программ? Как эти проблемы решают?

21. Разберитесь с особенностями переопределения в C++ следующих операций: присваивания `=`, `+=` и т. п.; индексирования `[]`; инкремента `++` и декремента `--`; косвенного выбора `->`; вызова функций `()`; преобразования типа `(type)`. Что такое функторы?

22. Найдите и самостоятельно изучите информацию об inline-функциях в C++.

23. Изучите исходный текст, приведённый в Листинге 5. Что вы можете сказать о принципе его работы? Что изменится (и почему?), если закомментировать строку 20 и раскомментировать строку 21?

```
1  from math import sqrt
2
3  class Complex:
4      def __init__(self, re=0, im=0):
5          self.re = re
6          self.im = im
7
8      def modulo(self):
9          return sqrt(self.re*self.re + self.im*self.im)
10
11     def __str__(self):
12         return "({0},{1})".format(self.re, self.im)
13
14     def __add__(self, other):
15         re = self.re + other.re
16         im = self.im + other.im
17         return Complex(re, im)
18
19 z1 = Complex(1, 2)
20 z2 = Complex(2, 3)
21 #z2 = 2
22
23 print(z1+z2)
```

```
24 print((z1+z2).modulo())
```

Листинг 5: Несложный пример работы с комплексными числами с позиций ООП на языке Python

24. Прочитайте официальную документацию о ссылках в языке Python. Что такое переменная в Python? Имеются ли в Python указатели?

25. Разберитесь с перегрузкой операций в языке Python.

26. Ознакомьтесь с переопределением (overriding) и перегрузкой (overloading) методов в Python. Как эти понятия соотносятся со статическим и динамическим полиморфизмом?

27. Изучите самостоятельно возможные методы обнаружения ошибок ввода при использовании объекта `std::cin`.

28. Почему не следует применять механизм обработки исключений для штатного выхода из вложенных управляющих конструкций или глубоких цепочек вызовов функций в программах на C++?

29. Приведите пример ситуации, когда функция `main` не сможет обработать возбуждённое исключение.

30. Что такое динамическая идентификация типов (RTTI — Run-Time Type Identification)? Какое это имеет отношение к обработке исключений?

31. Изучите самостоятельно особенности обработки исключений в языке Python.

32. Изучите рис. 1 совместно с листингом 6. Как можно уточнить данный рисунок? Действительно ли поле `a3` недоступно в классах-наследниках? Чем сокрытие отличается от приватности?

```
1 class A {
2 public:          // inherited with public, protected,
     private
3     int a1;
4 protected:       // inherited with public, protected,
     private
5     int a2;
6 private:         // not inherited
7     int a3;
8 };
9
```

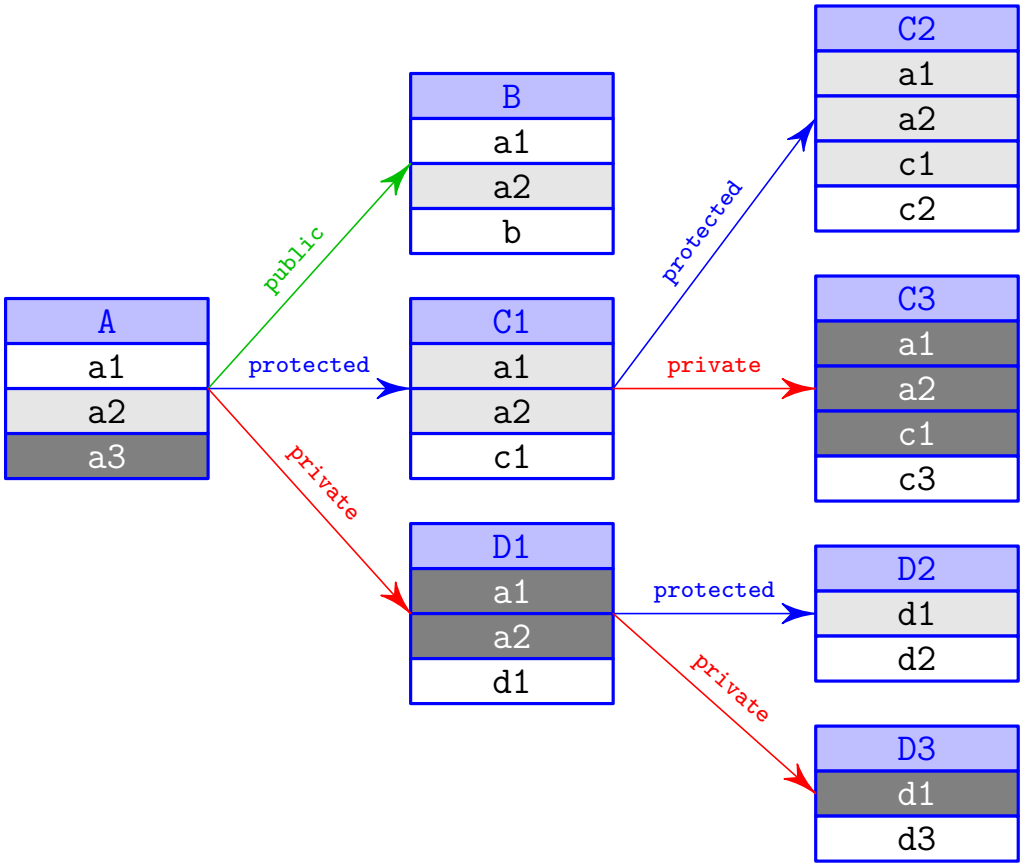

Рис. 1: Различные режимы наследования для трёх поколений классов

```
10 class B : public A {
11 public:
12     int b;       // In addition, it includes a1 as public
          and a2 as protected
13 };
14
15 class C1: protected A {
16 public:
17     int c1;      // In addition, it includes a1 and a2 as
          protected
18 };
19
20 class D1: private A {
21 public:
22     int d1;      // In addition, it includes a1 and a2 as
          private
23 };
```

Листинг 6: Различные режимы наследования

33. Изобразите диаграмму, аналогичную изображённой на рис. 2 для

класса `Circle`, унаследованного от класса `Pixel`? Как изменятся эти диаграммы при переходе к схеме наследования, основанной на абстрактном классе `GraphObject`?

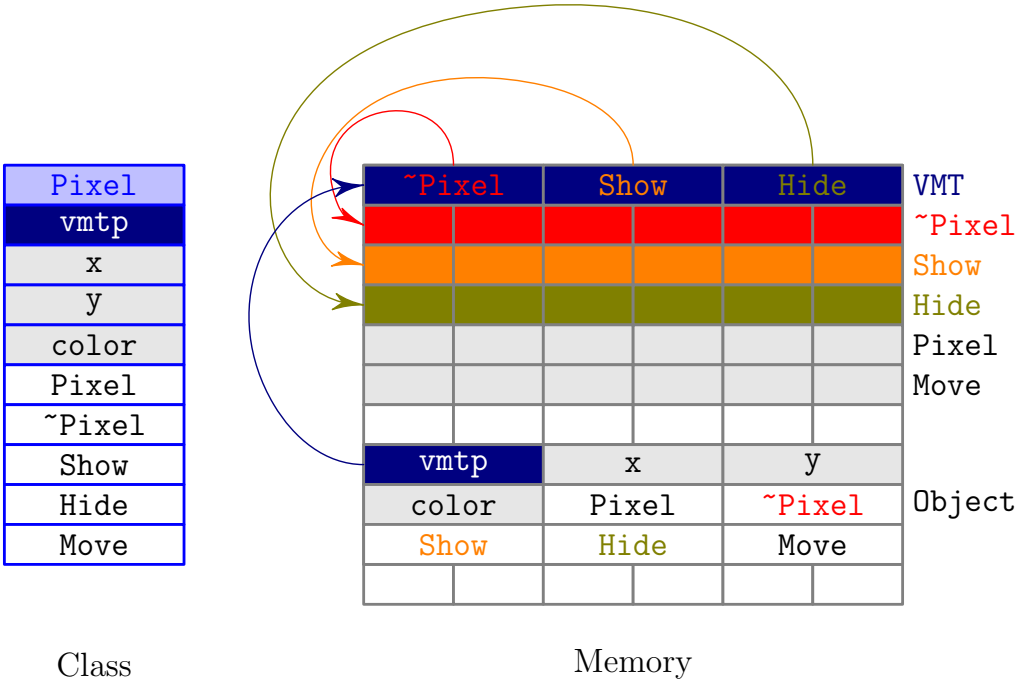

Рис. 2: Реализация идеи виртуальных методов на примере класса `Pixel`

34. Используя такие инструменты, как отладчик `gdb`, а также утилиты `hexdump` и `objdump`, исследуйте на практике реализацию механизмов наследования и виртуальных функций. Для этого напишите ряд несложных программ, использующих наследование, и изучите их машинный код: определите размеры родительского и наследуемых объектов; изучите структуру и адресацию полей и методов этих объектов; определите наличие у объектов поля `vtmp` и его свойства; зафиксируйте расположение в памяти и состав таблицы виртуальных методов.

35. Почему объект класса, имеющего приватный деструктор нельзя создать в виде простой (локальной или глобальной) переменной за пределами его методов и «друзей»?

36. Изучите самостоятельно концепцию множественного наследования. Какие у данного механизма преимущества и недостатки?

37. Почему функцию, приведённую в Листинге 7 нельзя использовать для сортировки массива элементов типа `double`? Можно ли это обойти, и как это сделать? К каким последствиям такое решение может привести?

```
1  void sort_int(int *array, int len) {
2      for(int start = 0; ; start++) {
3          bool done = true;
4          for(int i = len-2; i >= start; i--)
5              if(array[i+1] < array[i]) {
6                  int tmp = array[i];
7                  array[i] = array[i+1];
8                  array[i+1] = tmp;
9                  done = false;
10             }
11         if(done)
12             break;
13     }
14 }
```

Листинг 7: Сортировка «пузырьком» массива чисел типа `int`

38. По каким причинам решение “в лоб” с описанием функции `sort_double`, аналогичной функции `sort_int` Листинга 7, некорректно?

39. Прочитайте самостоятельно о многострочных макросах и макросах с параметрами препроцессора языка C. В чём их преимущества и недостатки? Можно ли использовать их в программах на языке C++? Если да, то рекомендуется ли это делать на практике и почему?

40. Каково предназначение шаблонов функций и механизма перегрузки функций? Действительно ли это одно и то же? Перечислите сходства и различия этих механизмов.

41. Обратите внимание на заголовок метода `Determinant`, приведённый в строке 14 Листинга 8, описывающего заголовок шаблона класса `QMatrix`. Приведите примеры ситуаций, когда использование такого шаблона метода некорректно. Продумайте возможные выходы из данных ситуаций.

```
1  template <class T> class QMatrix {
2      T** element;     // elements array
3      int N;           // matrix size
4  public:
5      QMatrix(int n = 0);
6      QMatrix(const QMatrix<T>&);
7      ~QMatrix();
8      int GetSize() const;
9      QMatrix<T> operator+(const QMatrix<T>&) const;
10     QMatrix<T> operator*(const QMatrix<T>&) const;
```

```
11      QMatrix<T> operator=(const QMatrix<T>&) const;
12      QMatrix<T> SubMatrix(int, int) const;    // minor
13      QMatrix<T> Inverse() const;              // inverse
           matrix
14      T Determinant() const;
15  };
```

Листинг 8: Шаблон класса `QMatrix`