1. **Перечислите все проблемы, которые вы видите в данном коде:**

class Foo

{

public:

Foo(int j) { i=new int[j]; }

~Foo() { delete i; }

private:

int\* i;

};

class Bar: Foo

{

public:

Bar(int j) { i=new char[j]; }

~Bar() { delete i; }

private:

char\* i;

};

void main()

{

Foo\* f=new Foo(100);

Foo\* b=new Bar(200);

\*f=\*b;

delete f;

delete b;

}

1. **Есть класс CodeGenerator, который умеет генерить код на разных языках.**

**Предложите рефакторинг с учетом, что количество языков будет расширяться**

class CodeGenerator

{

public:

enum Lang {JAVA, C\_PLUS\_PLUS, PHP};

CodeGenerator(Lang language) { \_language=language; }

std::string generateCode()

{

switch(\_language) {

case JAVA: //return generated java code

case C\_PLUS\_PLUS: //return generated C++ code

case PHP: //return generated PHP code

}

throw new std::logic\_error("Bad language");

}

std::string someCodeRelatedThing() // used in generateCode()

{

switch(\_language) {

case JAVA: //return generated java-related stuff

case C\_PLUS\_PLUS: //return generated C++-related stuff

case PHP: //return generated PHP-related stuff

}

throw new std::logic\_error("Bad language");

}

private:

Lang \_language;

}

1. **Все ли хорошо в этом коде?**

Файл legacy.c

int values[3];

Файл modern.cpp

#define LEGACY\_SIZE 3

extern int \*values;

class MyBlah {...};

class Adapter

{

public:

Adapter()

{

for (int i = 0; i < LEGACY\_SIZE; ++i)

map\_[values[i]] = new MyBlah (values[i]);

}

private:

std::map<int, MyBlah \*> map\_;

};

1. **Что не так в этом коде? Перечислите, какие недостатки вы видите. Предложите свой вариант рефакторинга.**

#include <stdio.h>

class Feature

{

public:

enum FeatureType {eUnknown, eCircle, eTriangle, eSquare};

Feature() : type(eUnknown), points(0) { }

~Feature()

{

if (points)

delete points;

}

bool isValid()

{

return type != eUnknown;

}

bool read(FILE\* file)

{

if (fread(&type, sizeof(FeatureType), 1, file) != sizeof(FeatureType))

return false;

short n = 0;

switch (type)

{

case eCircle: n = 3; break;

case eTriangle: n = 6; break;

case eSquare: n = 8; break;

default: type = eUnknown; return false;

}

points = new double[n];

if (!points)

return false;

return fread(&points, sizeof(double), n, file) == n\*sizeof(double);

}

void draw()

{

switch (type)

{

case eCircle: drawCircle(points[0], points[1], points[2]); break;

case eTriangle: drawPoligon(points, 6); break;

case eSquare: drawPoligon(points, 8); break;

}

}

protected:

void drawCircle(double centerX, double centerY, double radius);

void drawPoligon(double\* points, int size);

double\* points;

FeatureType type;

};

int main(int argc, char\* argv[])

{

Feature feature;

FILE\* file = fopen("features.dat", "r");

feature.read(file);

if (!feature.isValid())

return 1;

return 0;

}

1. **Имеем чистый C. Напишите generic функцию линейного поиска в массиве. И приведите пример использования вашей функции.**
2. **Имеем чистый С. Напишите reusable API для работы со связным списком структур. Структуры могу быть разного типа, но в одном списке будут структуры одного типа. Приведите пример использования вашего API.**