Wariant 1 2022

Nazwisko ……..

Imię ……..

Numer Indeksu ……..

N	Zadanie	Ocena

1	void main()
	{
	double a[11];
	float prec = 0.0001;
	float n = 2;
	//stosując *union*, umieścić w obszarze pamięci, zajmowanym zerowym
	//elementem tablicy a (a[0]), dwie zmienne: *prec* oraz *n*.
	
	fun(a);
	}
	
	void fun(double *aa)
	{
	//wyprowadzić na monitor zmienne *prec* oraz *n*, umieszczone w aa[0].
	
	}	

2	W jakich przypadkach programista powinien tworzyć konstruktor kopiujący? Dla podanego fragmentu kodu oznaczyć, co zobaczy użytkownik na monitorze.
	
	class A
	{
	public:
	A() { cout << “konstruktor\n”; }
	A(const A &) { cout << “konstruktor kopii \n”; }
	~A() { cout << “destructor\n”; }
	};
	
	A fun(A &c)
	{
	A b;	

	……
	return b;
	}
	
	int main()
	{
	A v;
	A ob = fun(v);
	return 0;
	}	

3	Czy podany fragment kodu jest poprawny? Jeśli nie, to proszę poprawić.
	
	`class ccc`
	`{`
	`int i;`
	`public:`
	`ccc(int ii) : i(ii) {}`
	`friend ostream & operator << (ostream &strm, const ccc &ob) {`
	`strm << ob.i;`
	`return strm;`
	`}`
	`};`
	
	`ccc * fun()`
	`{`
	`ccc i(10);`
	`return &i;`
	`}`
	
	int main()
	{
	`ccc *j = fun();`
	`cout << (*j);`
	`return 0;`
	}	

4.	Czy podany fragment kodu jest poprawny? Jeśli nie, to jak jego poprawić?
	
	class B
	{
	public:
	virtual void f() = 0;
	~B() {}
	};
	
	class D : public B
	{
	char *pchr;
	public:
	D(int dim) {	

	try {
	pchr = new char [dim];
	}
	catch(bad_alloc) {
	……..
	}
	}
	~D() { delete [] pchr; }
	void f() { cout << "jestem D\n"; }
	};
	
	int main()
	{
	B *pB = new D (10);
	pB->f();
	delete pB;
	return 0;
	}	

5.	Na czym polega polimorfizm dynamiczny? Podać przykład zastosowania operatora dynamic_cast().	

6.	Uzupełnić klasę B konstruktorem sparametryzowanym oraz przeciążyć operator = tak, żeby podany w funkcji main() fragment kodu działał poprawnie. Zmienna j powinna przyjmować wartość '+', jeśli i >= 0, i '-' w przeciwnym przypadku.
	
	class A
	{
	public:
	int i;
	public:
	A(int ii):i(ii){}
	A operator + (const A &ob) { A ret(0); ret.i = i + ob.i; return ret; }
	};
	
	class B : public A
	{
	char j;
	public:
	};	

	void main()
	{
	A ob1(1), ob2(-2);
	B ob3(0);
	ob3 = ob1 + ob2;
	}	

7.	class coord
	{
	double *cr; //cr[0] – x, cr[1] – y
	public:
	//Podaj deklaracji metod i operatorów jakie są potrzebny dla poprawnego działania
	//następującego kodu.
	
	};	

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

void main()
{
	coord aaa(0, 1), bbb(1, 0), ccc;
	ccc = sum(aaa, bbb);
}
```

8. Mamy hierarchie klas:

```
class A
{
	int a;
public:
	A(int aa) : a(aa) {}
	~A() {}
};

class B
{
	int b;
public:
	B(int bb) : b(bb) {}
	~B() {}
};

class C : public A, public B
{
	int c;
public:
	//stworzyć konstruktor sparametryzowany dla przekazywania danych zmiennym
	//a, b, c.




	~C() {}
};

void main()
{
	C ob(1, 2, 3);
	return;
```

	//w jakiej kolejności wywołują się konstruktory? Destruktory? }	