Agregace je vztah, kdy jeden objekt tvoří jeden celek, který jako části představuje jiné objekty.

- Pravda

Algoritmus je postup, který má následující vlastnosti:

- hromadný, deterministický, konečný

Automatické objekty se inicializují:

- Při vstupu do bloku.

Co to je čistě abstraktní třída?

- obsahuje pouze čistě virtuální metody.

Co je chybně v následujících deklaracích?

```
typedef struct Polozka {
   int klic;
   char *hodnota;
} Polozka;

const Polozka TAB[5];
```

- Konstantní objekty musí mít přiřazenou počáteční hodnotu

Co je chybně v následujícím fragmentu kódu:

```
void f(int x) { sem: return x+11; }
void g(float x) { x += 11; goto sem. }
```

- Nelze skákat z funkce do funkce

Co je zapotřebí doplnit do následující deklarace metody print, aby se jednalo o čistě virtuální metodu?

virtual void print()

- =0

Co vrací funkce rand z knihovny stdlib.h? Vyberte všechny správné odpovědi.

- Pseudonáhodné celé číslo v rozsahu 0..RAND_MAX
- Pseudonáhodné celé číslo.

Co vrátí funkce getenv("HOME") ze standardní knihovny stdlib.h?

- Obsah monitorové proměnné HOME - cesta k domovskému adresáři

Co vytiskne následující fragment kódu?

```
struct Bity {
   int :3;
   int val:4;
   unsigned int :9;
} x;
x.val = 011;
printf("%d\n", x.val);
```

- -7

Co vytiskne následující fragment kódu:

```
int x = 5;
printf("%d %d\n", x++, ++x);
```

- Výsledek nelze určit, závisí na implementaci (záleží na pořadí vyhodnocení parametrů)

Co vytiskne následující fragment kódu?

```
int n = 7;
switch ( n ) {
  case 6: printf("X"); break;
  case 7: printf("XX");
  case 8: printf("XXX"); break;
  default: printf("XXXX")
}
```

- XXXXX

Co způsobí následující fragment programu:

```
ifstream inFile("inputFile.txt");
inFile.fopen("inputFile.txt", ios::binary);
if (inFile.is_open()) {
  inFile.fseekg(0, ios::end);
}
```

- Pokud existuje soubor inputFile.txt nastaví ukazovátko pro čtení na jeho konec.

Číslo 174 lze v hexadekadické soustavě zapsat:

- AE, ae

Čistě abstraktní třída nemusí mít žádná členská data, ani žádné metody?

- Nepravda

Deklarovaná reference musí odkazovat na existující objekt?

- Pravda

Destruktorů může uživatel nadefinovat libovolný počet.

- Nepravda

Direktiva #include <soubor> zajistí vložení souboru do zdrojového textu:

- Místo řádku s direktivou

Direktivu #pragma musí preprocesor ve všech prostředích zpracovat podle definice?

- Nepravda

Doplňte následující deklaraci třídy D tak, aby byla čistě abstraktní

```
class D {
   int x;
public:
   virtual ~D() = ... ;
};
```

- 0

Dynamické proměnné v C++ vytváříme pomocí:

- Operátoru new

Funkce bsearch slouží pro řazení binárním půlením

- Nepravda

Funkce malloc z knihovny stdlib.h slouží pro:

- Slouží na alokaci místa ve volné paměti..

Funkce, které předznačíme jako inline, budou:

- Překládány jako makro nebo skok do podprogramu, rozhodne to překladač

Chráněná členská data a metody přístupné pouze pro potomky se označují klíčovým slovem:

- protected

Implicitní viditelnost položek ve struktuře (struct) je private?

- Nepravda

Implicitní způsob volání parametrů funkce je:

- Volání hodnotou

Inicializace podle seznamu inicializátorů se provádí:

- Podle pořadí položek v deklaraci

Jak lze zajistit při přetížení operátoru pomocí funkce přístup k datovým položkám třídy?

- Pomocí veřejných přístupových metod k položkám třídy (get)
- Označením funkce jako spřátelené (friend)

Jak se jmenuje speciální metoda objektu, která se volá výhradně při rušení objektu?

- Destruktor

Jak se jmenuje speciální metoda objektu, která se volá výhradně při vytvoření objektu?

- Konstruktor

Jak se jmenuje symbolická konstanta, která v knihovně limits.h definuje maximální celé číslo typu int?

- INT_MAX

Jak se nazývají proměnné, které nejsou dynamické?

- Statické

Jak se nazývá objekt (třída), který vzniká dědičností?

- Následovník
- Potomek
- Syn

Jak se v OOP nazývá objekt (třída), ze kterého děděním vznikají další objekty?

- Předchůdce
- Rodič

Jak se v OOP nazývá "proměnná", která je součástí objektu?

- Vlastnost
- Atribut
- Datová položka

Jak zjistíme velikost paměti, kterou zabírá proměnná x?

- Pomocí standardní funkce sizeof(x).

Jaká bude hodnota proměnné str po provedení následujícího kódu:

- Nazdar

Jaké jsou strukturované příkazy?

- Podmíněný příkaz, Přepínač, Cyklus

Jaké možnosti máme pro změnu hodnot prostřednictvím funkce?

- Použití ukazatelů jako parametrů
- Návratová hodnota funkce

Jaké typy konstruktorů existují? Vyberte smysluplné možnosti.

- Explicitní konstruktor – bezparametrický a parametrický
- Parametrický konstruktor
- Bezparametrický konstruktor

Jakého typu je výsledek výrazu: 'x' + 'y' ?

- int

Jako polymorfismus označujeme fakt, že stejné metody mohou znamenat různé věci u různých tříd:

- Pravda

Jakou hodnotu bude mít symbol EX po následující deklaraci výčtového typu:

```
enum {E1,E2=16,E3,E4=E3+5,EX];
```

- 23

Jakou hodnotu vrací knihovní funkce, které vrací ukazatel, v případě chyby?

- NULL

Jakou návratovou hodnotu používají zpravidla standardní funkce jako signalizaci správného chování?

- Hodnotu 0, pokud vrací hodnotu typu int

Jaký je rozdíl mezi strukturou a třídou?

- Ve třídě jsou položky implicitně privátní, ve struktuře veřejné.

Jaký je rozdíl mezi "třídou" a "objektem"?

- Objekt je instance třídy
- Třída popisuje vlastnosti objektů, objekt je pak reálný objekt dané třídy.

Jaký je vztah mezi výpočetními problémy a algoritmy?

- Pro každý výpočetní problém může existovat více algoritmů.
- Pro některé výpočetní problémy neexistuje žádný algoritmus.

Jaký vztah mezi algoritmem a programem platí?

- Každý algoritmus lze realizovat programem?

Jakým způsobem lze regulérně ukončit program jako správně provedený a dokončený?

- Příkazem return 0; ve funkci main
- Voláním funkce exit(0);

Jakými způsoby se řeší předávání informací o chybě z místa vzniku do místa obsluhy?

- Pomocí výjimek
- Pomocí globálních příznaků
- Pomocí návratových hodnot.

Jazyk C je podmnožinou jazyka C++?

- Nepravda

Jazyky C a C++ jsou neimperativní jazyky?

- Nepravda

Je dán uvedený příkaz: Data udaje;

- Třída se jmenuje Data a objekt se jmenuje udaje

Je dána definice třídy Data.

```
class Data
{
private:
    int d;

public:
    void set_d(int d) const
    {
        this->d = d;
    }

    int get_d() const
    {
        return d;
    }
};
```

- Nepravda

Je dána definice třídy Data

```
class Data
{
private:
   int d;

public:
   Data(int d)
   {
      set_d(d);
   }

   int get_d() const
   {
      return d;
   }

private:
   void set_d(int d)
   {
      this->d = d;
   }
};
```

Lze již vytvořený objekt této třídy měnit jeho stav?

- Nepravda

Je dána tato funkce/ metoda

Je dána tato funkce/metoda:

```
void m(int a, int b, int c = 4, int d = 5)
{
   ;
}
```

Napište volání metody (příkaz!) takový, aby se pro parametry *a* a *b* použily hodnoty proměnných *c* a *d*, a pro parametr *d* se použila hodnota 10. Nepoužívejte v zápisu mezery!!!

```
int c = ...;
int d = ...;
..... // volání funkce/metody main
```

-

Je dáno

```
class Data
{
    private:
        int d;
    public:
        void set_d(int d_new)
        {
            d = d_new;
        }

        int get_d() const
        {
           return d;
        }
};
```
Je správné použití příkazu Data data; ?

- Pravda

Je dáno

```
class Data
{
      private:
            int d;
      public:
            Data(int d)
            {
                  this->d = d;
            }

            void set_d(int d_new)
            {
                  d = d_new;
            }

            int get_d() const
            {
                 return d;
            }
};
```
Je správné použití příkazu Data data; ?

- Nepravda

Je následující zápis správný příkaz?

```
x = 7
```

- Nepravda

Je níže uvedené volání funkce f správné vzhledem k deklaraci?

```
void f(int);

f('a')
```

- Pravda

Je níže uvedené volání funkce f správné vzhledem k deklaracím?

```
void f(int, float);
void f(float, int);

f(1.2,2.1)
```

- Nepravda

Je správně a co znamená následující definice metody getObjem:

```
inline int Auto::getObjem() { return objem; }
```

- Je to správně, doporučení pro zvážení překladu jako makra.

Je správně následující definice konstruktoru třídy Auto?

```
class Auto{
private:
  int objem; string spz;
public:
  string getSpz () { return spz; }
  Auto(string sp, int obj);
};


Auto::Auto(string sp, int obj): objem(obj) { spz = sp); }
```

- Ano, je to správně

Je obecně správně následující deklarace, bez ohledu na místo, kde se vyskytuje:

```
const static int x;
```

- Nepravda

Je správně následující deklarace:

```
static const float x = 10;
```

- Pravda

Je správně následující příkaz, pokud je x deklarována jako proměnná typu int?

```
cin << x;
```

- Nepravda

Je správně následující příkaz, pokud je x deklarována jako proměnná typu int?

```
scanf("%d", x);
```

- Nepravda

Je správně uvedený fragment kódu definice třídy?

```
class Data
{
private:
    int d1;
    int d2;

public:
    Data() : Data(5)
    {
        ;
    }

    Data(int d) : Data(d, d)
    {
        ;
    }

    Data(int d1, int d2)
    {
        set_d1(d1);
        set_d2(d2);
    }

    void set_d1(int d1)
    {
        this->d1 = d1;
    }

    void set_d2(int d2)
    {
        this->d2 = d2;
    }

    ... ...
    ... ...
};
```

- Pravda

Je uvedená definice Data správná (kompilovatelná)?

```
class Data
{
private:
    int d;

public:
    Data(int d)
    {
        set_d(d);
    }

    int get_d() const
    {
        return d;
    }

private:
    void set_d(int d)
    {
        this->d = d;
    }
};
```

- Ano

Jsou následující fragmenty kódu ekvivalentní, za předpokladu deklarace int x;

```
printf("Hodnota: %d\n",x);


cout << "Hodnota: " << x << endl;
```

- Pravda

Jsou následující fragmenty kódu ekvivalentní, za předpokladu deklarace int x;

```
printf("Hodnota: %d\n",x);

printf("Hodnota: %d\n",x,x);
```

- Pravda

Kde se vytvářejí dynamické proměnné?

- Na haldě ve volné paměti.

Klíčové slovo static má následující důsledek:

- Takto označený objekt existuje po celou dobu výpočtu.

Kolik bytů zabírá konstanta ⟪literál⟫:

```
"gama\ndelta"
```

- 11

Kolik destruktorů může mít třída?

- Vždy právě jeden.

Kolik parametrů má níže uvedená metoda tisk?

```
class Datum {
  int rok; int mesic; int den;
public:
  void tisk(const char *poznamka);
};
```

- Má dva parametry, jeden implicitní ⟪this⟫ a jeden explicitní ⟪poznamka⟫

Kompozice je speciální případ agregace.

- Pravda

Konstruktor může být pro každou třídu vždy právě jeden.

- Nepravda

Která direktiva se používá pro vkládání hlaviček knihoven?

- #include

Která z následujících deklarací kopírujícího konstruktoru třídy Osoba je správná?

- Osoba(const Osoba&);

Která z následujících tvrzení jsou pravdivá?

- Pokud nedefinujeme destruktor, překladač jej vytvoří automaticky
- Destruktor se volá automaticky, nikoliv programově

Která z následujících knihoven je určena pro řešení chybových situací?

- errno.h

Které funkce slouží pro práci s binárními soubory?

- write, read

Které klíčové slovo můžeme použít místo klíčového slova typedef? (Pozor, záleží na velilkosti písmen!)

- using

Které klíčové slovo použijeme pro vyznačení statického objektu?

- static

Které knihovny používáme pro vstup a výstup v C/C++

- stdio.h, iostream, cstdio

Které kontejnery jsou součástí standardní knihovny šablon?

- Vektor, multimap

Které nástroje se používají pro popis algoritmů?

- Pseudokód, Vývojový diagram, Diagram aktivity UML

Které z následujících deklarací objektů třídy Osoba jsou správně?

- Osoba o;, Osoba o{};, Osoba* o = new Osoba;, Osoba* o = new Osoba();, Osoba* o = new Osoba{};

Které z následujících konstruktů se používají pro zpracování chyb?

- Výjimky, Návratové hodnoty, Globální příznaky

Které z následujících objektů tvoří standardní prostředí programu v C

- stdin, stdout, stderr

Které z následujících operací označujeme jako tzv. speciální funkce (operace)?

- Destruktor
- Základní konstruktor
- Kopírující konstruktor
- Přesunující přiřazení
- Přesunující konstruktor
- Kopírující přiřazení

Které z následujících postupů se nepoužívají pro řešení chybových situací?

- Chybové situace se nedají řešit
- Lze ponechat na řešení kompilátorem

Které z následujících stylů se používají při řešení problémů?

- Objektově – orientovaný
- Strukturovaný

Kterým klíčovým slovem požádáme překladač o doplnění typu proměnné? (Pozor, záleží na velikosti písmen!)

- auto

Logické výrazy se povinně vyhodnocují zprava do leva?

- Nepravda

Lze ignorovat chybu, při které je generována výjimka?

- Nepravda

Lze ignorovat chyby hlášené pomocí návratových hodnot?

- Pravda

Lze operátor <<přetížit metodou?

- Nepravda

Lze v C++ přidávat zcela nové operátory?

- Nepravda

Lze v C++ přetěžovat operátory jako členské metody?

- Většinu lze přetěžovat i jako metodu, některé ale nikoliv

Má lambda funkce definované jméno?

- Nepravda

Mějme následující deklarace:

```
double calc(double);
int count(const string &, char);

Která z následujících volání se nepřeloží?

(a) calc(23.4, 55.1);
(b) count("abcda", 'a');
(c) calc(66);
```

- Výraz (c) je správně.
- Výraz (b) je správně.

Mějme následující deklarace:

```
double calc(double, double);
int count(char *, char);
(a) calc(23.4, 55.1);
(b) count("abcda", 'a');
(c) calc(66);
```

- Výraz (c) se nepřeloží.

Modifikující operátor se vyznačuje tím, že mění svůj argument.

- Pravda

Může být konstruktor virtuální metoda?

- Nepravda

Může být spřátelená funkce virtuální?

- Nepravda

Může být v jazyce C++ zapsán tento fragment kódu?

```
void m()
{
    ;
}

void m(int a, int b)
{
    ;
}
```

- Pravda

Může být v jazyce C++ zapsán tento fragment kódu?

```
void m()
{
    ;
}

void m(int a, int b)
{
    ;
}

void m(int a, int b, int c = 4, int d = 5)
{
    ;
}
```

- Nepravda

Může mít abstraktní třída instance?

- Nepravda

Najděte chyby v následující deklaraci struktury:

```
struct osoba {
  static char *jmeno;
  char *prijmeni;
  extern int vek;
  float plat;
}
```

- Klíčové slovo extern udávající paměťovou třídu se může použít jen na celou strukturu

Najděte chyby v následujícím kódu:

```
#include<iostream>

#include<string>

int main() {

 std::string str{ "Hello!" };

 for(auto c : str) { std::cout << "[" << c << "]"; }

  std::cout << '\n';

}
```

- Je to správný kód

Napište deklaraci destruktoru pro třídu TCosi (nezapomeňte deklaraci řádně ukončit).

- ~TCosi();

Napište deklaraci kopírujícího konstruktoru třídy T, kterou zakážeme, aby jej generoval překladač

- T(const T&) delete;

Napište deklaraci kopírujícího konstruktoru třídy T (pozor na malá a velká písmena, nepoužívejte nadbytečné mezery). Parametr neoznačujte jménem, neuvádějte středník na konci deklarace.

- T(const T&)

Napište klíčové slovo, které můžete použít místo těla kopírujícího konstruktoru, pokud chcete, aby za Vás tento konstruktor vytvořil překladač?

- default

Napište klíčové slovo, kterým aktivujete dynamickou vazbu metody v deklaraci třídy (na velikosti písmen v odpovědi záleží):

- virtual

Napište název základní třídy pro definici výjimek:

- exception

Napište příkaz, který je třeba doplnit do níže uvedeného kódu, aby se vstupní soubor správně uzavřel a uvolnil všechny alokované prostředky.

```
ifstreamifile;
ifile.open("heslo.txt");
if(!ifile.is_open()) { cout << "soubor nelze otevrit"; }
else{ cout << "OK"; }
```

- ifile.close();

Napište volání metody, kterou zařídíte přesun objektu x do objektu stejného typu.

- std::move(x)

Novou verzi operátoru << pro výstup informací do datového proudu můžeme vytvořit jako:

- Jako přátelský operátor
- Jako operátor, pokud jsou k dispozici metody typu get

Objekty jsou vytvářeny pomocí konstruktorů?

- Pravda

Obsahuje následující funkce chybu?

```
void square(int i, int *y) {
  int pow;
  pow = i * i;
  y = &pow;
}
```

- Není – vrací referenci na zrušený objekt.

Ovladač catch(...) lze použít kdekoliv mezi ovladači?

- Nepravda

Ovladače výjimek označujeme klíčovým slovem catch.

- Pravda

Po indikaci chyby přes globální příznak platí:

- Může nastat více chyb, ale indikovat lze jen poslední.
- Je nutno chybu hned indikovat.
- Nelze zjistit, kde chyba vznikla.

Pokud chceme v definici virtuální metody zdůraznit, že se skutečně jedná o jinou realizaci metody, jaké klíčové slovo použijeme v hlavičce (pozor na velká a malá písmena)?

- override

Pokud potřebujeme, aby se volání metody přizpůsobilo dynamicky typu objektu, nad kterým se provádí, musíme:

- metodu označit jako virtuální klíčovým slovem virtual

Pokud v deklaraci třídy neuvedeme kopírující konstruktor, jaká z následujících tvrzení jsou správná?

- Vždy se vytvoří standardní kopírující konstruktor ﴿vytvoří mělkou kopii﴾

Pokud vytvoříme třídu B pomocí veřejné dědičnosti za třídy A:

```
class B : public A {...}
pak můžeme:
```

- při realizaci metod třídy B používat veřejné metody třídy A
- pro objekty třídy B používat veřejné metody třídy A

Pořadí vyhodnocení parametrů funkce je:

- Je dáno implementací, nikoliv jazykem

Používá se v jazyce v C++ pojem „přetížení funkcí“ a „přetížení metod“?

- Pravda

Pro formátovaný vstup a výstup v C/C++ používáme funkce

- printf, scanf

Pro převod čísel z dekadické soustavy do soustavy o základu Z lze použít algoritmus, kdy převáděné číslo dělíme základem a vždy zbytek po dělení zapíšeme jako výstupní číslici. Zbytek po dělení je totiž správná cifra v soustavě o základu Z.

- Pravda

Pro umožnění práce s vnějšími soubory, které netvoří standardní prostředí musíme použít které z následujících funkcí

- Konstruktory tříd iostream
- fopen
- fclose
- open
- close

Pro třídu Data:

```
class Data
{
private:
    int d;

public:
    Data()
    {
        set_d(5);
    }

    Data(int d)
    {
        set_d(2 * d);
    }

    int get_d() const
    {
        return d;
    }

    void set_d(int d)
    {
        this->d = d;
    }
};
```

Je vytvořen objekt *data2* podle následující deklarace:

```
Data data2(15);
```

Jaká hodnota je poskytnuta výrazem *data2.get_d()* ?

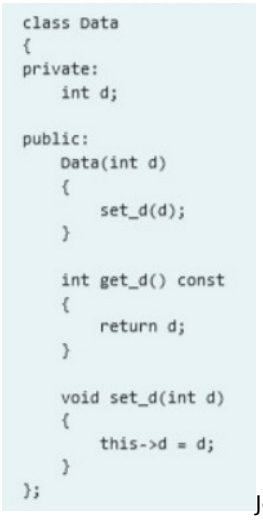
- 30

Pro uvedenou definici třídy:

```
class Data
{
private:
    int d;

public:

    Data(int d)
    {
        this->d = d;
    }

    int get_d() const
    {
        return d;
    }

    int value() const
    {
        return d * 4;
    }

    void calc(int v)
    {
        d = v * 0.25;
    }

private:
    void set_d(int d)
    {
        this->d = d;
    }

};
```

- Data(int d), get_d0), value(), calc(int v)

Pro uvedenou třídu Data



```
class Data
{
private:
    int d;

public:
    Data(int d)
    {
        set_d(d);
    }

    int get_d() const
    {
        return d;
    }

    void set_d(int d)
    {
        this->d = d;
    }
};
```

Je správný zápis vytvoření objektu třída?

- Nepravda

Pro vstup a výstup v jazyce C se používají příkazy read a write?

- Nepravda

Přátelskou funkci zavedeme pomocí klíčového slova "mate"?

- Nepravda

Předpokládejte následující deklaraci funkce fn.

```
void fn (int n, int a = 0, char *ret = "tisk");
```

Je správné následující volání?

```
fn(11,"ahoj")
```

- Nepravda

Přetížíme-li operátor jako funkci, jak zajistíme přístup k datovým položkám třídy?

- Pomocí definice operátoru jako spřátelené funkce.
- Pomocí metod typu get modifikované třídy.

Při použití dědičnosti může proměnná typu ukazatel na základní třídu ukazovat i na potomky této třídy?

- Ano, neboť všichni potomci mají i data ze základní třídy.

Při řešení chyb pomocí globálních příznaků není možné ignorovat chybu? (pravda = není možné)

- Nepravda

Při řešení chyb pomocí návratové hodnoty:

- Je možné indikovat jen jednu chybu.

Příkaz goto lze použít pouze v rámci funkce?

- Pravda

Standardní třída exception požaduje při použití definici metody, která slouží pro indikaci chybové situace. Jak se tato metoda jmenuje?

- what

Statické datové položky třídy jsou součástí každé instance?

- Nepravda

Statické metody třídy mohou:

- používat jen statické metody této třídy
- používat jen statické datové položky této třídy

Statické objekty se inicializují při startu programu?

- Pravda

Stav objektu je dán:

- Stavem jeho atributů

Struktura nemůže mít žádný konstruktor?

- Nepravda

Třída musí mít vždy právě jeden konstruktor?

- Nepravda

Třída může mít jen jeden konstruktor?

- Nepravda

Třída může mít libovolný počet destruktorů?

- Nepravda

Určete chyby v následujícím fragmentu kódu:

```
struct TCosi {
private:
  int hodnota = 1;
public:
  getHodnota(void);
};
```

```
struct TCosi x;
```

```
x.hodnota = 7;
printf("%d\n", x.getHodnota());
```

- Položky nemohou mít implicitní hodnoty.
- Přímý přístup k položkám private není povolen

Určete chyby v následujícím fragmentu kódu:

```
class TCosi {
  int hodnota;
  int pocet;
public:
  TCosi(int h, int p) { hodnota = h; pocet = p; }
  int getHodnota(void) { return hodnota; };
  void setHodnota(int h) { hodnota = h; };
};

TCosi x(7,2);

cout << x.getHodnota() << endl;
```

- Je to správně

Určete chyby v následujícím fragmentu kódu:

```
struct TCosi {
  int hodnota;
  int pocet;
public:
  TCosi(int h, int p) { hodnota = h; pocet = p; }
  int getHodnota(void) { return hodnota; };
  void setHodnota(int h) { hodnota = h; };
};

struct TCosi x(7,2);

printf("%d\n", x.getHodnota());
```

- Je to správně

Určete chyby v následujícím programu: **// Ještě 1 ta samá otázka, ale místo return – exit(0)**

```
#include <iostream>
#include <fstream>

int main() {
  ofstream soubor("data.txt");
  soubor << "Vstupní text\n";
  return 0;
}
```

- Chyba – soubor není uzavřen // stejná odpověď

Určete chyby v následujícím programu:

```
#include <stdio.h>

int main() {
  FILE *soubor = fopen("data.txt","r");
  fprintf(soubor, "Nejaky text\n");
  return 0;
}
```

- Chyba – soubor není uzavřen
- Chyba – soubor nemá určen mód pro zápis "w"

Určete všechny chyby v následujícím fragmentu kódu:

```
class TCosi {
  int hodnota = 1;
public:
  getHodnota(void);
};

TCosi x;

x.hodnota = 7;
cout << x.getHodnota() << endl;
```

- Položky nemohou mít implicitní hodnoty
- Přímý přístup k položkám private není povolen

Určete výstup následujícího fragmentu kódu:

```
class TCosi {
  int hodnota;
  int pocet;
public:
  TCosi(int h, int p) { hodnota = h; pocet = p; }
  int getHodnota(void) { return hodnota; };
  void setHodnota(int h) { hodnota = h; };
};

TCosi x(7,2);

std::cout << x.getHodnota();
x.setHodnota(8);
std::cout << x.getHodnota();
```

- 78

Uvažte fragment kódu:

```
int x = 1;
if ((x > 2) && (x = 0)) x = 2;
```

Jaká bude hodnota proměnné x po vyhodnocení logického výrazu v podmínce?

- 1

Uvažte následující deklaraci třídy String:

```
class String {
  int pocet;  // aktuální počet znaků v řetězci
  char *text; // ukazatel na C-kovský řetězec
public:
. . .
};
```

Je nutno vytvářet explicitní destruktor pro tuto třídu?

- Pravda

Uvažte následující fragment kódu

```
int x = 3;
int y = 4;
if ((x = y) && (x == y)) x = 15;
```

Jaká bude hodnota proměnné x:

- 15

Uvažte následující fragment kódu

```
#include <cerrno> // errno
#include <string.h> // strerror
#include <stdio.h> // prinf, FILE, fopen
int main() {
  FILE *f = fopen("moje.data","r");

  if (f == NULL) { printf("Chyba: %d\n", errno); }
  return 0;
}
```

Co bude na standardním výstupu po jeho skončení, pokud soubor moje.data neexistuje (číslo této chyby je 2)?

- 2

Uvažte následující fragment kódu:

```
string str;
```

Je správně zapsána následující hlavička příkazu cyklu, kterým se projdou všechny znaky řetězce str:

```
for (auto c: str) ... c ...
```

- Pravda

Uvažte následující fragment kódu:

```
int x = 7;
x = x << 3;
```

Jaká bude hodnota proměnné x?

- 56

Uvažte následující fragment kódu:

```
int a[10];
int i = 3;
a[i] = i++;
```

Jaká bude hodnota prvků a[3] a a[4]?

- Výsledek nelze určit, závisí na implementaci (záleží na pořadí vyhodnocení ++ a =)

Uvažte následující fragment kódu:

```
char str[20];
```

Je správně zapsána následující hlavička příkazu cyklu, kterým se projdou všechny znaky pole str:

```
for (auto c: str) ... c ...
```

- Pravda

Uvažujte následující fragment programu. Co bude na standardním výstupu po jeho provedení?

```
int i, &ri = i;
int *pi = &i;
i = 5; ri =10;
*pi = 15;
std::cout << i << " " << ri << std::endl;
```

- 15 15

Uvažte následující fragment kódu. Která níže uvedená tvrzení jsou pravdivá?

```
int a = 0, b = 0, c = 1;

void funkce(int a, int b) {
   int a = -5;
   cout << "main: a = " << a << ", b = " << b << ", c = " << c << endl;
}
```

- Chybná deklarace proměnné a

Uvažte následující fragment kódu. Která níže uvedená tvrzení jsou pravdivá?

```
int a = 0, b = 0, c = 1;

void funkce(int a, int b) {
   printf("main: a = %d, b = %d, c = %d\n", a, b, c);
}
```

- Je to správný kód

Uvažujem definici třídy DataA

```
class DataA
{
private:
   int a;

public:
   DataA()  { a = 50;  }
   DataA(int a)  { this->a = 2 * a; }
   void SetA(int a) {
      if (a > 0) this->a = a;
   }
   int GetA() const { return a; }
   int Calc() const { return a * 3; }
};
A příkazy programu
   DataA o1(20);
   int a = o1.GetA();
Jaká je hodnota proměnné a po vykonání příkazů?
```

- 40

Uvažujem definici třídy DataA.

```
class DataA
{
private:
   int a;

public:
   DataA() { a = 50; }
   DataA(int a) { this->a = 2 * a; }
   void SetA(int a) {
      if (a > 0) this->a = a;
   }
   int GetA() const { return a; }
   int Calc() const { return a * 3; }
};
A příkazy programu
   DataA o3(10);
   o3.SetA(20);
   int d = o3.GetA();

Jaká je hodnota proměnné d po vykonání příkazů?
```

- 20

Uvažujem definici třídy DataA

```
class DataA
{
private:
   int a;

public:
   DataA() { a = 50; }
   DataA(int a) { this->a = 2 * a; }
   void SetA(int a) {
      if (a > 0) this->a = a;
   }
   int GetA() const { return a; }
   int Calc() const { return a * 3; }
};
A příkazy programu
   DataA o2;
   int b = o2.Calc();
   int c = o2.GetA();

Jaká je hodnota proměnných b a c po vykonání příkazů?
```

- b ... 150; c ... 50

Uvažujem definici třídy DataA a DataB.

```
class DataA {
private:
   int a;

public:
   DataA() { a = 50; }
   DataA(int a) { this->a = 2 * a; }
   void SetA(int a) {
      if (a > 0) this->a = a;
   }
   int GetA() const { return a; }
   int Calc() const { return a * 3; }
};
class DataB : public DataA {
public:
   DataB() : DataA(5) { }
};
A příkazy programu
   DataB o4;
   int x = o4.GetA();

Jaká je hodnota proměnné x po vykonání příkazů?
```

- 10

Uvažujme definici třídy DataA umístěnou v souboru DataA.h.

Jak vypadá fragment, kterým si zajistíme možnost použití třídy v hlavním programu, v souboru Main.cpp?

- #include "DataA.h"

Uvažujte následující fragment programu. Co bude na standardním výstupu po jeho provedení?

```
int i, &ri = i;
int *pi = &i;
i = 5; ri =10;
*pi = 15;
std::cout << i << " " << ri << std::endl;
```

- 15 15

Uvažte třídu T a operátor přiřazení = s deklarací:

```
T& operator=(const T& p){ . . . };

Jak v těle definice operátoru = poznáme, že přiřazovaný objekt není totožný s tím, kterému jsme žádost o přiřazení adresovali?
```

- Pomocí testu: if (this == &p) return *this;

Uvažujeme třídu Data, jak se nazývá uvedený příkaz?

- Deklarace objektu

V C++ lze zadávat implicitní hodnoty parametrů:

- Vždy jen odzadu

V C může existovat více funkcí stejného jména? Přesněji: mohou se v C přetěžovat funkce?

- Nepravda

V objektově-orientovaném programování, jaké známe principy_programování?

- Zapouzdření; Dědičnost; Polymorfizmus

Viditelnost položek ve struktuře nebo třídě lze upřesnit klíčovými slovy:

- Public, Private, Protected

Vyberte z možností pravdivé charakteristiky šablonových kontejnerů:

- Kontejner list → je obousměrně zřetězený seznam

- Kontejner deque → sekvenční kontejner do kterého lze přidávat zepředu i zezadu
- Kontejner map → asociativní kontejner realizující zobecněné pole
- Kontejner vector → poskytuje dynamické pole
- Kontejner array → realizuje pole pevné délky
- Kontejner set → realizuje asociativní kontejner typu množina

Vyberte, který příkaz by měl být správně doplněn místo ---

```
int main () {
  int *pp = new int[10];
  for (int i = 0; i < 10; i++) pp[i] = i;
   ---
  return 0;
}
```

- Je třeba doplnit delete [] pp;

Vyberte správné tvrzení:

- Objekt má vždy alespoň jeden konstruktor.

Výjimky jsou identifikovány:

- Typem

Zápis:

```
int f(int x, float y);
```

je definice funkce f?

- Nepravda

Zápis 0123 je:

- Zápis celého čísla 83

Zapište dekadicky hodnotu hexadekadického čísla 1AE.

- 430

Zapište klíčové slovo, kterým je možné výjimku generovat?

- throw

Zapište klíčové slovo, kterým v těle metod třídy označujeme ukazatel na objekt, se kterým metoda pracuje.

- this

Zaškrtněte všechny správné možnosti vstupu a výstupu v C/C++.

- Textový standardní vstup a výstup
- Binární vstup a výstup
- Textový vstup a výstup