Programowanie obiektowe - Przykładowe zadania egzaminacyjne (2005/2006)

Część 1. Teoria

- Wyjaśnij pojęcia, podaj przykład:
  - klasa
  - obiekt
  - konstruktor
  - destruktor
  - kapsułkowanie (hermetyzacja)
  - wskaźnik this
  - zaprzyjaźnianie
  - przeciążanie
  - przestrzeń nazw
  - wyjątki
  - referencja
  - wskaźnik
  - zmienna statyczna
  - funkcja statyczna

- Które z poniższych stwierdzeń *najlepiej* opisuje funkcję konstruktora klasy?
  A. Testuje wszystkie funkcje składowe klasy.
  B. Inicjalizuje dane składowe obiektu klasy.
  C. Określa i zwraca ilość pamięci potrzebnej dla danych składowych klasy.
  D. Zwalnia pamięć zajętą przez obiekt klasy.
  E. Wyświetla informację o utworzeniu nowego obiektu klasy.

- Konstruktor charakteryzuje się pewnymi cechami. Które z podanych poniżej stwierdzeń jest *nieprawdziwe*?

  A. Konstruktor jest wywoływany automatycznie, gdy deklarowana jest nowa zmienna.
  B. Jeśli klasa nie ma własnego konstruktora, kompilator dostarczy konstruktor domyślny.
  C. Konstruktor nie może być składową prywatną.
  D. Konstruktor może nie mieć żadnych parametrów.
  E. Konstruktor zwraca typ utworzonego obiektu.

- Mamy trzy klasy, co do których wiadomo, że:

  I. Klasa `Punkt2w` zawiera dwie składowe typu `float`.
  II. Konstruktor klasy Wektor dynamicznie przydziela pamięć na składowe wektora.
  III. W klasie `Lista_Płac` każdy obiekt musi mieć jednoznacznie przydzielony numer identyfikacyjny.

  Dla której klasy powinniśmy utworzyć konstruktor kopiujący?

A. tylko I B. tylko II C. tylko III D. II i III E. I i II

- Nazwij mechanizm, który musi być zaimplementowany w klasie `MojaKlasa`, aby można było wykonać następujące działania:

A.	B.	C.

`MojaKlasa a;`
`...`
`cout << a;`
 	
`.....`
	`MojaKlasa a,b;`
	`...`
	`b=a;`
	
	`.....`
		`TwojaKlasa a;`
		`...`
		`MojaKlasa b=a;`
		
		`.....`

- W którym z poniższych fragmentów użyty będzie konstruktor kopiujący?

```
A. Data d1;
B. Data d1("Luty",15,2001);
C. Data d1; d1.UstawDate("Luty",15,2001);
D. Data d1, d2; d1.UstawDate("Luty",15,2001); d2=d1;
E. Data d1("Luty",15,2001); Data d2=d1;
```

- Dana jest klasa Tekst pozwalające przechowywać napisy. Jakie konstruktory będą potrzebne w tej klasie, aby możliwe było wykonanie następujących operacji? Podaj prototypy tych funkcji.

```
Tekst a="Jezyk C";
Tekst ca=a;
const Tekst b="C++";
Tekst cb=b;
```

- Które z poniższych stwierdzeń najlepiej opisuje funkcję destruktora klasy?
  A. Usuwa wartości wszystkich danych składowych klasy.
  B. Blokuje dostęp do funkcji składowych klasy.
  C. Wyświetla komunikat o błędzie i zatrzymuje program.
  D. Zwraca pamięć dynamicznie przydzieloną danym składowym klasy do puli pamięci dostępnej dla programu.
  E. Przypisuje wszystkim składowym typu wskaźnikowego wartość NULL.

- Każde z poniższych stwierdzeń uzupełnij odpowiednim słowem z zestawu <*zawsze*, *czasem*, *nigdy*>
  A. Destruktory ..... pobierają argumenty.
  B. Destruktor ..... występuje kilka razy w definicji klasy.
  C. Destruktory ..... dynamicznie zwalniają pamięć za pomocą operatora `delete`.
  D. Konstruktory ..... pobierają argumenty.
  E. Konstruktory ..... zwracają wynik.
  F. Konstruktory ..... dynamicznie przydzielają pamięć za pomocą operatora `new`.

- Prawda czy fałsz? Przy każdym z poniższych stwierdzeń zaznacz P lub F.
  A. Funkcje składowe zawsze są deklarowane za pomocą prototypu, zaś ich definicja umieszczana jest na zewnątrz klasy.
  B. Funkcje składowe klasy zdefiniowane w ciele klasy automatycznie mają nadawany status funkcji `inline`.
  C. Aby funkcja składowa klasy otrzymała status funkcji `inline`, zawsze musi być poprzedzona słowem `inline`.
  D. Składową klasy może być inna klasa.
  E. Składowa statyczna może być inicjalizowana wewnątrz klasy.
  F. Metoda statyczna ma dostęp tylko do składowych publicznych klasy
  G. Deklaracja friend może wystąpić w dowolnym miejscu definicji klasy.
  H. Jeśli klasa A jest klasą zaprzyjaźnioną klasy B, wtedy B jest zawsze zaprzyjaźnione z A.

- Które z poniższych stwierdzeń dotyczących funkcji przeciążających operatory jest *nieprawdziwe*:
  A. Można przeciążać tylko operatory wbudowane języka C++.
  B. Można przeciążać operator tylko wtedy, kiedy jednym z argumentów jest typ własny użytkownika.
  C. Liczba argumentów operatora nie może być zmieniana.
  D. Przeciążanie nie zmienia priorytetu operatorów.
  E. Można definiować nowe operatory z operatorów istniejących w języku, np. `**` dla potęgowania.

- Podaj przykłady, kiedy trzeba wykorzystać wskaźnik, nie można skorzystać z referencji.

- Wskaż błąd oraz opisz w jaki sposób można go poprawić.

```
class P {
private:
  int dana;
  static int licznik;
public:
  P(int y=5) { dana =y; }
  int ZwiekszDane() const {return ++dana; }
  static int PobierzLicznik()
  {
    cout << "Dane to: " << dana << endl;
    return licznik;
  }
};
```

- Dana jest definicja:

```
struct Demo
{
  int iDemo;
};
Demo obDemo;
Demo *wskDemo = &obDemo;
```

Które z poniższych wyrażeń pozwalają na dostęp do składowej `iDemo`?

```
A. obDemo.iDemo
B. (*obDemo).iDemo
C. obDemo->iDemo
D. wskDemo.iDemo
E. (*wskDemo).iDemo
```

- Co wydrukuje poniższy program:

```
#include <iostream>
int i=1;
int main() {
  cout << i << ',';
  {
     int i=2;
     cout << i << ',';
     ::i=3;
     cout << i << ',';
     i=::i;
     cout << i << ',';
     ::i=4;
  }
  cout << i;
  cout << endl;
  char z;
  cin >> z;
}
```

- Opisz zachowanie następującego programu:.

```
#include <iostream>
using namespace std;
class Punkt {
  double x,y;
public:
  double PobierzX() {return x;}
};
int main()
{
  Punkt p;
  double x,y;
  x=100;
  cout << PobierzX() << endl;
  return 0;
}
```

A. Wystąpi błąd w fazie kompilacji	B. Wystąpi błąd w fazie linkowania	C. Program wyświetli 0	D. Program wyświetli 100	E. Wynik może być różny w każdym przebiegu

Jeśli program nie jest poprawny jak go zmienić?

- Opisz zachowanie następującego programu. Jeśli program nie jest poprawny jak go zmienić?

```
#include <iostream>
using namespace std;
class Dane {
  double a,b;
public:
  double DrukujA() {return a;}
};
int main()
{
  Dane d;
  double a,b;
  a=50;
  cout << DrukujA() << endl;
  return 0;
}
```

A. Program wyświetli 0	B. Program wyświetli 50	C. Wystąpi błąd w fazie kompilacji	D. Wystąpi błąd w fazie linkowania	E. Wynik może być różny w każdym przebiegu

- Załóżmy, że dana jest definicja:

```
class Ksiazka {
  double cena;  // cena książki
public:
  ...          // konstruktory
  double Cena() const;  // zwraca cenę książki
};
```

Chcemy napisać funkcję, która jest klientem klasy `Ksiazka`:

```
double Suma(Ksiazka zestaw[], int ile)
// funkcja zwraca sumę cen książek z tablicy zestaw
{
  int k;
  double razem=0.0;
  for (k=0; k<ile;k++)
    instrukcje
  return razem;
}
```

Którą z instrukcji należy uzupełnić podaną funkcję:

A. `razem += zestaw.Ksiazka[k];`
B. `razem += zestaw.Ksiazka.Cena(k);`
C. `razem += zestaw[k].Cena();`
D. `razem += zestaw[k].Ksiazka();`
E. `razem += zestaw[k].Ksiazka.Cena();`

- Załóżmy, że dana jest definicja klasy:

```
class Ksiazka {
    string tytul;
    int l_egz;   // liczba egzemplarzy aktualnie dostępnych
    int l_stron; // liczba stron w książce
    ...
  public:
    ...
    int PodajLiczbeEgz(); // zwraca liczbę dostępnych egzemplarzy
    int PodajLiczbeStron(); // zwraca liczbę stron w książce
    ...
};
```

Uzupełnij funkcję, która jest klientem klasy `Ksiazka`. Do funkcji tej przekazywana jest tablica zawierająca katalog książek. Każda pozycja katalogu opisuje jeden tytuł.

```
int SumaStron(const vector<Ksiazka> & katalog) {
// zwraca sumę stron we wszystkich książkach aktualnie dostępnych
int k;
  int suma=0;
  for (k=0; k<katalog.length();k++)
  { instrukcje }
  return suma;
}
```

Uwaga: Funkcja `int length() const` zwraca rozmiar wektora.

- Dana jest funkcja:

```
bool CechaX(const vector<int> &v) {
  bool flaga=false;
  int i;
  for (i=0; i<v.lenght()-1; i++)
    flaga = flaga || (v[i] == v[i+1]);
  return flaga;
}
```

Jak najlepiej opisać działanie tej funkcji? (Funkcja `int length() const` zwraca rozmiar tablicy).

A. Zwraca `true`, jeśli elementy tablicy `v` są posortowane wzrastająco, w przeciwnym wypadku zwraca `false`.
B. Zwraca `true`, jeśli elementy tablicy `v` są posortowane malejąco, w przeciwnym wypadku zwraca `false`.
C. Zwraca `true`, jeśli tablica `v` ma dwa elementy o tej samej wartości, w przeciwnym wypadku zwraca `false`.
D. Zwraca `true`, jeśli wszystkie elementy tablicy `v` mają różne wartości, w przeciwnym wypadku zwraca `false`.
E. Zwraca `true`, jeśli tablica `v` ma dwa elementy o tej samej wartości umieszczone obok siebie, w przeciwnym wypadku zwraca `false`.

- Załóżmy, że dana jest następująca definicja klasy Lista:

```
class Lista{
  struct WezelListy{
    int dane;
    WezelListy *nast;
  };
  WezelListy *pierwszy; // pierwszy węzeł listy
  WezelListy *ostatni;  // ostatni węzeł listy
public:
  Lista(); // konstruktor
  void DodajNaKoniec(int k); // dopisz k na koniec listy
};
void Lista::DodajNaKoniec(int k)
{
  WezelListy *roboczy = new WezelListy;
  roboczy->dane = k; roboczy->nast=NULL;
  if (ostatni != NULL) ostatni->nast = roboczy;
  ostatni=roboczy;
}
```

Funkcja `DodajNaKoniec` nie działa prawidłowo. Które z poniższych stwierdzeń opisuje sytuację, w której funkcja *nie działa prawidłowo*:

A. Zawsze działa nieprawidłowo
B. Jeśli lista jest początkowo pusta
C. Jeśli lista nie jest już pusta.
D. Jeśli lista zawiera jedną wartość.
E. Jeśli lista zawiera już *k* wartości.

- Załóżmy, że dane są funkcje:

```
void F(int &a, int &b) {
  a++;
  b++;
}
int main() {
  int liczba=1;
  F(liczba,liczba);
  cout << liczba << endl;  return 0;
}
```

Jaki wynik zobaczymy na ekranie:

A. `0` B. `1` C. `2` D. `3` E. Komunikat o błędzie

- Załóżmy, że dane są funkcje:

```
void Modyfikuj(int x, int &y, int &z)
{x=y; y=z; z=x; }
void Przetwarzaj(int a, int b, int c) {
  Modyfikuj(a,b,c);
  cout << a <<" " << b << " " << c << endl;
}
```

Funkcja `Przetwarzaj()` wywoływana jest następująco: `Przetwarzaj(1,2,3)`. Jaki wynik zobaczymy na ekranie:

A. `1 2 3` B. `2 1 3` C. `1 3 2` D. `2 3 1` E. `3 2 1`

- Dana jest funkcja:

```
int F(int &x, int &y)
{ x *= 2; y *=2; return x*y; }
```

Co zostanie wyświetlone w wyniku wykonania instrukcji:

```
int a=3;
cout << F(a,a) << ' ';
cout << a << endl;
```

A. 9 3 B. 18 3 C. 36 6 D. 39 12 E. 144 12

- Która deklaracja umożliwi wykonanie następujących działań:

```
const Klasa K;
cout<<K<<endl;
```

A.
```
class Klasa   {
 char N;
 friend ostream &operator<<(ostream &out,const Klasa &K);
};
ostream &operator<<(ostream &out,const Klasa &K) { return out<<K.N; }
```

B.
```
class Klasa   {
 unsigned N;
 friend ostream &operator<<(ostream &out,Klasa &K);
};
ostream &operator<<(ostream &out,Klasa &K) { return out<<K.N; }
```

C.
```
class Klasa   {
 int N;
 ostream &operator<<(ostream &out,const Klasa &K) { return out<<K.N; }
};
```

- Która deklaracja umożliwi wykonanie następujących działań:

```
Klasa K;
++K;
```

A.
```
class Klasa   {
 public:
 signed char N;
 Klasa &operator++() { N++; return *this; }
};
```

B.
```
class Klasa   {
 public:
 int N;
};
Klasa &operator++(Klasa &K) { ++K.N; return K; }
```

C.
```
class Klasa   {
 public:
 short N;
 Klasa &operator++(int) { ++N; return *this; }
};
```

- Która deklaracja umożliwi wykonanie następujących działań:

```
Klasa K,X=K;
```

A.
```
class Klasa {
  public:
  int N;
  Klasa(const Klasa &K):N(K.N) {}
 };
```

B.
```
class Klasa   {
  public:
  unsigned char N;
 };
```

C.
```
class Klasa
 {
  public:
  int N;
  Klasa(int n):N(n) {}
  Klasa &operator=(const Klasa &K) { N=K.N; }
 };
```

Część 2: Zadania

Zadanie 1.

Zdefiniować klasę `PostepGeometryczny` (przykłady: 2,4,8,16,...; 1,-3,9,-27,81,...
wzory: $a_n = a_1 q^{(n-1)}$)
według następujących wymagań:

a) funkcja składowa `Element(int a)` zwraca *n*-ty wyraz postępu; n=0 oznacza wyraz początkowy
b) obiekty mogą być tworzone na następujące 4 sposoby:
`PostepGeometryczny p0, p1(3), p2(1,2), p(p2);`
gdzie pierwszy parametr oznacza wyraz początkowy, drugi iloraz postępu, brak parametrów oznacza postęp od 1 z ilorazem 2
c) funkcja składowa `Suma(int n)` zwraca sumę *n* pierwszych wyrazów postępu ($S_n = a_1 (q^n-1)/(q-1)$)
d) funkcja składowa `Drukuj(int od, int po)` wyprowadza do strumienia `cout` elementy o numerach `od..do`.

Zaproponować funkcję `main()` do przetestowania opracowanej klasy.

Zadanie 2.

Klasa `stack` (stos) oprócz konstruktorów zawiera następujące funkcje:
`void push(char element)` - wkłada dany element na stos
`void pop()` - pobiera element ze stosu
`void pop(char & element)` - pobiera element ze stosu do parametru element
`char top() const` - zwraca element znajdujący się na wierzchołku stosu, bez usuwania go ze stosu
`bool empty() const` - zwraca `true`, jeśli stos jest pusty

Napisać funkcję zewnętrzną `OdwrocStos`. Funkcja ta ma tworzyć i zwracać stos zawierający te same elementy co dany stos S, ale ułożone w odwrotnym porządku. Po zakończeniu działania funkcji stos S ma pozostać niezmieniony.

Zadanie 3.

Napisać funkcję `Usun`, która usuwa napisy z posortowanej tablicy. Funkcja ma działać następująco:
- jeśli napisu nie ma w tablicy, nie należy nic robić,
- jeśli napis jest w tablicy, należy go usunąć i zawartość tablicy przesunąć w lewo o jedno miejsce oraz zmniejszyć `ile` o 1.

Nagłówek opracowywanej funkcji ma postać:

```
void Usun(const string & slowo, vector <string> & wykaz, int & ile)
// slowo zawiera usuwany napis
// wykaz jest tablicą napisów posortowanych alfabetycznie
// ile określa rozmiar tablicy,
//     zakładamy, że 0 <= ile < wykaz.lenght()
```

Uwaga:
Klasa `vector` ma przeciążony operator `[]`, który pozwala pobrać element o wskazanym indeksie.
Klasa `string` pozwala porównać alfabetycznie dwa napisy za pomocą przeciążonego operatora == , jeśli lewy napisy jest równy prawemu, zwracana jest wartość `true`; w przeciwnym wypadku zwracana jest wartość `false`.

Zadanie 4.

Załóżmy, że dane o studentach są przechowywane w listach dowiązaniowych. Każda grupa posiada swoją listę:

```
struct Student {
  string nazwisko;
  double srednia; // średnia ocena
  Student *nast;
  Student(const string nazw, double sr, Student * n); // konstruktor
};
struct Grupa {
  string nazwa_gr;  // nazwa grupy
  Student *lista_gr; // lista studentów w grupie
  Grupa();     // konstruktor
  Grupa(const string & nazwa);    // konstruktor
...
};
```

Napisz funkcję, która będzie zliczała studentów ze średnimi powyżej określonej wartości w podanej grupie:

```
int LiczSrednie(const Grupa & jakas, double ocena)
```