Katedra
Informatyki i Automatyki
Politechniki Rzeszowskiej

Informatyka

5.Programowanie w C++

Opracował: Maciej Penar

Spis treści

1. Zagadnienia teoretyczne

Odpowiedzieć na pytania / zaimplementować programy:

1. Jaka jest różnica pomiędzy strukturą a klasą
2. Jaka jest różnica pomiędzy klasą a obiektem
3. Dana jest następująca klasa:

```
class some_class {};
```

Jakie operatory/metody/moduły zostaną utworzone automatycznie przez kompilator?

4. Jaka jest różnica pomiędzy modyfikatorem dostępu private/protected/public. Jaka jest rola zaprzyjaźniania? Jakie konstrukcje języka mogą być poddane zaprzyjaźnieniu?
5. Dlaczego ten kod działa:

```
class test {
public:
     std::string _holder;
     test(char value[]) : _holder(value) {};
};

int main()
{
     test t = "abba"; // czemu to działa?
     std::cout << t._holder << std::endl;
}
```

6. Wyjaśnić na czym polega dziedziczenie / polimorfizm.
7. ~~Zaproponować klasę **complex** reprezentującą liczbę zespoloną:~~
   ~~https://pl.wikipedia.org/wiki/Liczby_zespolone~~
   ~~Zaimplementować oraz pokazać działanie następujących części składowych zaproponowanej klasy:~~
   a. ~~Napisać metodę zwracającą moduł liczby~~
   b. ~~Przeciążyć operator + dla wartości typu double (dodawanie do części rzeczywistej)~~
   c. ~~Przeciążyć operator + dla wartość/referencji typu complex (części rzeczywiste do rzeczywistych, urojone do urojonych).~~
   d. ~~Przeciążyć operator * dla wartości typu double (mnożenie części rzeczywistej i urojonej)~~
   e. ~~Przeciążyć operator * dla wartości/referencji typu complex (patrz. Podpowiedź)~~

~~Podpowiedź do 5~~
~~Odnośnie liczb zespolonych. Dla zadania istotny jest fakt że składają się one z dwóch wektorów: **rzeczywistego** oraz **urojonego**. Część urojona z definicji jest przemnożona przez wartość urojoną i. Wartość urojona ma ciekawą własność $i^2 = -1$~~ ~~Modułem nazywamy długość wektora wypadkowego: $\|m\| = \sqrt{Re^2 + Im^2}$~~ ~~Mnożenie liczb zespolonych jest o tyle podchwytliwe ze względu na $i^2 = -1$ np.~~ ~~$(5 - i) * (2 + 3i) = 10 - 2i + 15i - 3i^2 = 10 + 13i + 3 = 13 + 13i$~~

8. Jaka jest różnica (jeśli istnieje) pomiędzy przeciążaniem operatorów wewnątrz klasy, a poza nią? Jeśli istnieje różnica, napisać program ilustrujący ją (teraz to już wiadomo że jest). Patrz przykład:

Przeciążanie poza klasą:

```
class some_class { .. };

some_class operator+(some_class& t_first, some_class& t_second) { ... };
```

Przeciążanie wewnątrz klasy:

```
class some_class {
public:
```

```
        some_class operator+(some_class& t_second) {...};
};
```

9. Co to jest singleton. Przedstawić jego implementacje.

2. Najbardziej istotne zadanie na tej liście z C++

Poniżej przedstawiono kod złej implementacji klasy String (kod znajduje się na repo). Przeanalizować poniższy program oraz wyjście z konsoli. Zidentyfikować błędy oraz poprawić/wskazać jak poprawić. (Język C++, S. Prata, roz. 12, str. 554).

```
#include <iostream>
#include <cstring>
#include <windows.h> // Dla znaków na konsoli Windowsa

typedef char character;
class StringBad {
private:
        character * str;
        int len;
        static int num_strings;
public:
        StringBad(const character * s) {
                len = std::strlen(s);
                str = new character[len + 1];
                std::strcpy(str, s);
                num_strings++;

        std::cout << num_strings << ": \"" << str << "\" - obiekt utworzony.\n";
        }
        StringBad() {
                len = 4;
                str = new character[4];
                std::strcpy(str, "C++");
                num_strings++;

        std::cout << num_strings << ": \"" << str << "\" - obiekt domyślny utwor
zony.\n";
        }
        ~StringBad() {
                std::cout << "\"" << str << "\" - obiekt usunięty,";
                --num_strings;
                std::cout << "są jeszcze " << num_strings << ".\n";
                delete[] str;
        };
        friend std::ostream & operator<<(std::ostream & os, const StringBad & st
) {
                os << st.str;
                return os;
        }
};

int StringBad::num_strings = 0;

void callVal(StringBad val) {
        std::cout << "Obiekt przekazany przez wartość:" << std::endl;
        std::cout << "\"" << val << "\"" << std::endl;
}
```

```
void callRef(StringBad & val) {
        std::cout << "Obiekt przekazany przez referencje:" << std::endl;
        std::cout << "\"" << val << "\"" << std::endl;
}

int main()
{
        /*
        Wymagane dla polskich znaków na Windows
        */
        SetConsoleOutputCP(65001);
        setlocale(LC_ALL, "65001");

        using std::cout; using std::endl;
        {
                cout << "Zaczynamy blok" << endl;
                StringBad msg1("Witaj na kursie Informatyka");
                StringBad msg2("Programujemy w C");
                StringBad msg3("Kiedyś będziemy uczyć się sieci");

                cout << "Pierwsza wiadomość: " << msg1 << endl;
                cout << "Druga wiadomość: " << msg2 << endl;
                cout << "Trzecia wiadomość: " << msg3 << endl << endl;

                callRef(msg1);

        cout << "Pierwsza wiadomość raz jeszcze: " << msg1 << endl << endl;

                callVal(msg2);

        cout << "Druga wiadomość raz jeszcze: " << msg2 << endl << endl;


        cout << "Spróbujmy coś innego, inicjalizacja innym obiektem: " << endl;
                StringBad evenWorse = msg3;

        cout << "Trzecia wiadomość z innej zmiennej: " << evenWorse << endl << e
ndl;


        cout << "Ciśniemy dalej: przypisanie do innego obiektem: " << endl;
                StringBad worst;
                cout << "I teraz przypisanie!" << endl;
                worst = msg1;

        cout << "Pierwsza wiadomość z innej zmiennej: " << worst << endl << endl
;


                cout << "I wychodzimy z bloku" << endl;
        }
        cout << "Koniec main";
        return 0;
}
```

Wynik z CodeBlocks / Visual Studio:

```
Zaczynamy blok
1: "Witaj na kursie Informatyka" - obiekt utworzony.
```

2: "Programujemy w C" - obiekt utworzony.
3: "Kiedyś będziemy uczyć się sieci" - obiekt utworzony.
Pierwsza wiadomość: Witaj na kursie Informatyka
Druga wiadomość: Programujemy w C
Trzecia wiadomość: Kiedyś będziemy uczyć się sieci

Obiekt przekazany przez referencje:
"Witaj na kursie Informatyka"
Pierwsza wiadomość raz jeszcze: Witaj na kursie Informatyka

Obiekt przekazany przez wartość:
"Programujemy w C"
"Programujemy w C" - obiekt usunięty,są jeszcze 2.
Druga wiadomość raz jeszcze:
8

Spróbujmy coś innego, inicjalizacja innym obiektem:
Trzecia wiadomość z innej zmiennej: Kiedyś będziemy uczyć się sieci

Ciśniemy dalej: przypisanie do innego obiektem:
3: "C++" - obiekt domyślny utworzony.
I teraz przypisanie!
Pierwsza wiadomość z innej zmiennej: Witaj na kursie Informatyka

I wychodzimy z bloku
"Witaj na kursie Informatyka" - obiekt usunięty,są jeszcze 2.
"Kiedyś będziemy uczyć się sieci" - obiekt usunięty,są jeszcze 1.
"r

8

---

będziemy uczyć się sieci" - obiekt usunięty,są jeszcze 0.
<- Tu Visual wyrzuca wyjątek
"C++" - obiekt usunięty,są jeszcze -1.

" - obiekt usunięty,są jeszcze -2.
Koniec main

3. Programowanie w C++

Wybrać program 1) lub 2) i napisać w C++:

1. Zaprojektować i zaimplementować grę w kółko i krzyżyk na planszy 3x3.
    a. Każdy ruch powinien być drukować aktualny stan planszy na konsoli.
    b. Program powinien wspierać grę:
        i. Przeciwko drugiemu graczu
        ii. Przeciwko AI (niech losuje ruch)
    c. Przemyśleć projekt, przeanalizować, zoptymalizować. Na kolejnych listach chciałbym byście zrealizowali sieciową wersję. Jeśli dobre zaprojektujecie, to „usieciowienie” gry sprowadzi się do rozwiązania problemu wymiany komunikatów pomiędzy komputerami.

2. Zaprogramować tzw. grę w życie (link: https://pl.wikipedia.org/wiki/Gra_w_%C5%BCycie). Czyli alokujemy tablicę $n\ x\ m$ znaków {'o' (martwa),'x' (żywa)}. Założyć pewną liczbę iteracji np. 100. Zaimplementować tzw. reguły Conwaya.

Reguły „Gry w życie" Conwaya
• Martwa komórka, która ma dokładnie 3 żywych sąsiadów, staje się żywa w następnej jednostce czasu (rodzi się)[1][2][3] • Żywa komórka z 2 albo 3 żywymi sąsiadami pozostaje nadal żywa; przy innej liczbie sąsiadów umiera (z „samotności" albo „zatłoczenia"

Jeśli potraficie, to wyabstrahować następujące elementy:

- Ustalanie stanu początkowego (*** nie jest takie proste**), dwie realizacje
  - Wczytanie z pliku (pokazać dowolny oscylator np. szybowiec)
  - Losowanie
- Reguły przeżywalności komórki, dwie realizacje
  - Reguły Conwaya
  - Wymyślić swoje własne

4. Algorytmika

Opisać rozwiązanie lub zaimplementować w jakimkolwiek języku. W C++ do wykonania zadania wystarczą następujące nagłówki:

```
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
```

Źródło wiadome – jak ktoś był uważny wie skąd.

Michał przygotowuje się do zawodów programistycznych na które składają się N testów próbnych. Michał wierzy w zabobony. Wierzy że może „zachować szczęście". Każdy test próbny opisuje dwoma liczbami całkowitymi L_i oraz T_i

- L_i jest to liczba szczęścia powiązana z testem próbnym. Jeśli Michał **podejdzie** do testu próbnego, to jego szczęście opada o wartość L_i. Jeśli **nie podejdzie** do testu próbnego to odkłada szczęście, czyli zwiększa je o wartość L_i.
- T_i oznacza ważność testu próbnego. Jeśli jest równa 1 to test jest **ważny,** jeśli jest równy 0 to test **nie jest ważny**

Dane wejściowe: Pierwsza linia w pliku to wartość dwie wartości całkowite oddzielone spacją:

$N - liczba\ testów\ próbnych$

$K - liczba\ ważnych\ konkursów\ które\ Michał\ może\ przegrać$

Potem wystąpi N linii, gdzie $i - ta$ linia składa się z dwóch wartości całkowitych oddzielonych spacją: L_i oraz T_i

Ograniczenia

- $1 \leq \mathrm{N} \leq 100$
- $0 \leq K \leq N$
- $1 \leq L_i \leq 10^4$
- $0 \leq T_i \leq 1$

Wyjście

Wypisać maksymalną możliwą do uzyskania przez Michała liczbę szczęścia.

Przykład:

Wejście	Wyjście
6 3 5 1 2 1 1 1 8 1 10 0 5 0	29

Liczba testów próbnych to $N = 6$, z czego 4 są ważne. Michał może nie wziąć udziału w trzech testach próbnych $K = 3$.

Michał bierze udział tylko w trzecim teście próbnym o liczbie szczęścia $L_3 = 1$. Oznacz to że szczęście Michała to: $5 + 2 - 1 + 8 + 10 + 5 = 29$