Programowanie obiektowe (TZ1E2010)

Politechnika Białostocka - Wydział Elektryczny

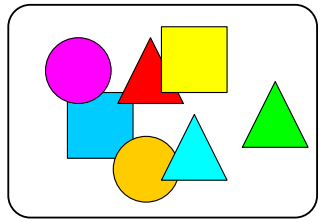
Elektronika i telekomunikacja, semestr II
studia niestacjonarne I stopnia

Rok akademicki 2020/2021

Pracownia nr 8 (14.05.2021)

dr inż. Jarosław Forenc

Funkcje wirtualne (polimorfizm)



Przykład

- program ma wyświetlać na ekranie różne figury (kwadrat, trójkąt, koło)
- do wyświetlenia każdej figury stosowana jest inna funkcja, figury powinny być wyświetlane na ekranie w określonej kolejności

Problem: Jak zorganizować przechowywanie informacji o figurach? Jak zorganizować wyświetlanie figur?

Rozwiązanie: Klasy + dziedziczenie + funkcje wirtualne

- definiujemy klasę podstawową (figura) oraz trzy klasy pochodne (kwadrat, trojkat, kolo)
- w klasie podstawowej umieszczamy funkcję void rysuj() poprzedzoną słowem virtual (funkcja ta nic nie robi)
- w klasach pochodnych umieszczamy funkcje o takich samych nazwach jak w klasie podstawowej - void rysuj() wyświetlające poszczególne figury

Przykład: funkcje wirtualne

```
#include <iostream>
using namespace std;

class figura
{
   public:
     virtual void rysuj() { };
};

class kwadrat : public figura
{
   public:
     void rysuj()
     {
         cout << "Rysuj: kwadrat" << endl;
     }
};
```

klasa podstawowa figura

funkcja wirtualna rysuj()

klasa pochodna kwadrat

Przykład: funkcje wirtualne

```
class trojkat : public figura
{
   public:
     void rysuj()
     {
         cout << "Rysuj: trojkat" << endl;
     }
};

class kolo : public figura
{
   public:
     void rysuj()
     {
         cout << "Rysuj: kolo" << endl;
     }
};
```

klasa pochodna trojkat

klasa pochodna kolo

Funkcje wirtualne (polimorfizm)

- jeśli wskaźnikowi do klasy podstawowej (figura) przypiszemy adres obiektu klasy pochodnej (kwadrat, trojkat, kolo), to wywołując poprzez wskaźnik funkcję rysuj(), wywołamy funkcję odpowiadającą danemu obiektowi, np.

  figura *ptr; - deklaracja wskaźnika do obiektu klasy figura
  kwadrat kw1; - deklaracja obiektu klasy kwadrat
  trojkat tr1; - deklaracja obiektu klasy trojkat
  kolo kol1; - deklaracja obiektu klasy kolo

  ptr = &kw1;
  ptr->rysuj(); - wywołana zostanie funkcja rysuj() z klasy kwadrat

  ptr = &tr1;
  ptr->rysuj(); - wywołana zostanie funkcja rysuj() z klasy trojkat

  ptr = &kol1;
  ptr->rysuj(); - wywołana zostanie funkcja rysuj() z klasy kolo

- mówimy, że w powyższym przykładzie wystąpił polimorfizm (wielopostaciowość)

Przykład: funkcje wirtualne

```
int main(void)
{
    kwadrat kwadrat1, kwadrat2;
    trojkat trojkat1, trojkat2;
    kolo    kolo1, kolo2;
    figura  *lista[6];

    lista[0] = &trojkat1;
    lista[1] = &kwadrat1;
    lista[2] = &kolo1;
    lista[3] = &kwadrat2;
    lista[4] = &kolo2;
    lista[5] = &trojkat2;

    for (int i=0; i<6; i++)
       lista[i]->rysuj();
}
```

```
Rysuj: trojkat
Rysuj: kwadrat
Rysuj: kolo
Rysuj: kwadrat
Rysuj: kolo
Rysuj: trojkat
```

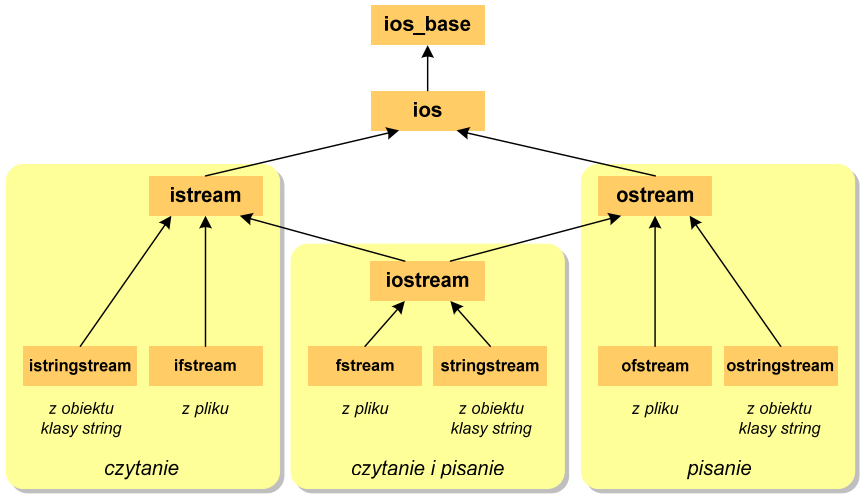

Obsługa plików w języku C++

- do przetwarzania plików w języku C++ stosowane są strumienie zrealizowane w postaci klas
- do podstawowych klas należą:
  - ofstream (output file stream) - klasa zapewniająca zapis do pliku
  - ifstream (input file stream) - klasa zapewniająca odczytywanie pliku
  - fstream (file stream) - klasa zapewniająca zapis i odczytywanie pliku
- zastosowanie powyższych klas wymaga dołączenia w programie pliku nagłówkowego fstream

```
#include <iostream>
#include <fstream>
using namespace std;
```

- nazwy zadeklarowane w tym pliku wchodzą w skład przestrzeni nazw std

Hierarchia klas strumieni w języku C++



Obsługa plików w języku C++

- zazwyczaj operacje związane z przetwarzaniem pliku składają się z czterech etapów:

1. Zdefiniowanie strumienia czyli stworzenie obiektu jednej z klas: ifstream, ofstream, fstream.
2. Otwarcie pliku (określenie z jakim plikiem strumień ma komunikować się).
3. Wykonanie operacji na pliku.
4. Zamknięcie pliku (zlikwidowanie strumienia).

Otwarcie pliku (strumienia)

- strumień można otworzyć albo za pomocą konstruktora danej klasy albo wywołując jej funkcję składową open()
- konstruktor i funkcja open() mają dokładnie takie same argumenty

```
void open(const char *fname, ios_base::openmode mode);
```

- fname - nazwa pliku z ewentualną ścieżką dostępu (C-string)
- mode - tryb pracy z plikiem (rodzaj dostępu do pliku):
  - in (input) - otwarcie pliku do czytania
  - out (output) - otwarcie pliku do pisania
  - ate (at end) - otwarcie pliku i ustawienie na jego końcu
  - app (append) - otwarcie pliku do dopisywania na jego końcu
  - trunc (truncate) - otwarcie pliku i skasowanie aktualnej zawartości
  - binary (binary) - otwarcie pliku w trybie binarnym (domyślnie jest tryb tekstowy)

Otwarcie pliku (strumienia)

```
ofstream ofs;
ofs.open("dane.txt");
```

- wywołanie funkcji składowej open() z klasy ofstream w celu otwarcie pliku dane.txt w trybie tekstowym (domyślny tryb) do zapisu (ios::out - domyślny tryb dla klasy ofstream)

```
ifstream ifs("dane.txt");
```

- zastosowanie konstruktora klasy ifstream w celu otwarcie pliku dane.txt w trybie tekstowym (domyślny tryb) do odczytu (ios::in - domyślny tryb dla klasy ifstream)

Otwarcie pliku (strumienia)

```
ifstream ifs;
ifs.open("D:\\doc\\dane.dat",ios::in | ios::binary);
```

- otwarcie pliku dane.dat znajdującego się na dysku D: w folderze doc do odczytu (ios::in) w trybie binarnym (ios::binary)

```
ofstream ofs;
string fname("D:\\doc\\dane.txt");
ofs.open(fname.c_str(),ios::out);
```

- otwarcie pliku dane.txt znajdującego się na dysku D: w folderze doc do zapisu (ios::out) w trybie tekstowym (domyślny tryb)
- funkcja c_str() zwraca tekst przechowywany w obiekcie klasy string w postaci tablicy znaków typu char * (C-string)

Zamknięcie pliku (strumienia)

- zamknięcie pliku odbywa się poprzez wywołanie metody close()

```
void close();
```

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
   ofstream ofs;

   ofs.open("liczby.txt");
   if (!ofs)
   {
       cout << "Blad otwarcia pliku!" << endl;
       return 0;
   }
   // operacje na pliku
   ofs.close();
}
```


Operacje na pliku tekstowym

- klasa ofstream jest pochodną klasy ostream, klasa ifstream jest pochodną klasy istream, zaś klasa fstream jest pochodną klasy iostream
- powyższe oznacza, że wszystko co dotyczyło strumieni iostream (manipulatory, operatory << i >>) dotyczy także operacji na plikach:
  - flush, endl
  - hex, dec, oct
  - showbase, noshowbase
  - showpos, noshowpos
  - showpoint, noshowpoint
  - fixed, scientific
  - setprecision(n), setw(n), setfill(znak)

Przykład: zapisanie liczb do pliku tekstowego

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <ctime>
using namespace std;

#define N 4
#define M 5

int main()
{
    float T[N][M];
    int i, j;
    ofstream ofs;

    ofs.open("liczby.txt");
    if (!ofs)
    {
       cout << "Blad otwarcia pliku!" << endl;
       return 0;
    }
```


Przykład: zapisanie liczb do pliku tekstowego

```
    srand((unsigned int)time(NULL));

    for (i=0; i<N; i++)
       for (j=0; j<M; j++)
          T[i][j] = 100*(float)rand()/RAND_MAX - 50;

    ofs << fixed << setprecision(3);

    for (i=0; i<N; i++)
    {
       for (j=0; j<M; j++)
          ofs << setw(10) << T[i][j];
       ofs << endl;
    }

    ofs.close();
}
```

```
     7.045   -31.448   -15.273    26.254     5.126
   -12.590    46.115    16.543    25.945     4.769
   -10.051    -8.721     1.180    26.244    47.464
   -37.515    27.624    12.868    31.323   -45.111
```

Przykład: odczytanie liczb z pliku tekstowego

```
#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;

int main(void)
{
   float x;
   ifstream ifs;

   ifs.open("liczby.txt");
   if (!ifs)
   {
       cout << "Blad otwarcia pliku!" << endl;
       return 0;
   }

   cout << fixed << setprecision(3);
   while (ifs >> x)
      cout << setw(10) << x << endl;

   ifs.close();
}
```

```
  7.045
-31.448
-15.273
 26.254
  5.126
-12.590
 46.115
 16.543
 25.945
  4.769
-10.051
 -8.721
  1.180
 26.244
 47.464
-37.515
 27.624
 12.868
 31.323
-45.111
```


Operacje na pliku binarnym

```
istream & read(char *s, streamsize n);
```

- odczytuje ze strumienia n bajtów i umieszcza je w tablicy s
- jeśli odczytywany jest obiekt innego typu niż char, to należy dokonać odpowiedniego rzutowania do typu char *

```
ostream & write(const char *s, streamsize n);
```

- pobiera z tablicy s n bajtów i wstawia je do strumienia
- jeśli operacja nie może się udać, to zamiast referencji do strumienia, na którym pracuje, zwraca wartość zero (0)
- jeśli mamy obiekt innego typu niż char, to należy dokonać odpowiedniego rzutowania do typu char *

Operacje na pliku binarnym

- do pliku binarnego zapisywana jest wartość zmiennej x1 typu float, a następnie plik jest zamykany
- po ponownym otwarciu pliku, zapisana w nim liczba jest odczytywana i wyświetlana na ekranie

```
float x1 = 100, x2;

ofstream ofs("dane.dat",ios::binary);
ofs.write((char *) &x1,sizeof(x1));
ofs.close();

ifstream ifs("dane.dat",ios::binary);
ifs.read((char *) &x2,sizeof(x2));
ifs.close();

cout << "x1 = " << x1 << " x2 = " << x2 << endl;
```


Przykład: odczytanie liczb z pliku binarnego

```
int main(void)
{
   float x;
   ifstream ifs;

   ifs.open("liczby.dat", ios::in | ios::binary);
   if (!ifs)
   {
       cout << "Blad otwarcia pliku!" << endl;
       return 0;
   }

   cout << fixed << setprecision(3);

   ifs.read((char *)&x,sizeof(float));
   while(!ifs.eof())
   {
       cout << setw(10) << x << endl;
       ifs.read((char *)&x,sizeof(float));
   }
   ifs.close();
}
```

```
  7.045
-31.448
-15.273
 26.254
  5.126
-12.590
 46.115
 16.543
 25.945
  4.769
-10.051
 -8.721
  1.180
 26.244
 47.464
-37.515
 27.624
 12.868
 31.323
-45.111
```