Język C++ zajęcia nr 1

I. Programowanie obiektowe

Zasadniczą cechą programowania obiektowego jest łączne rozpatrywanie zagadnień dotyczących algorytmów i struktur danych. Wyrazem tego jest zmiana w sposobie reprezentacji rzeczywistych obiektów przejawiająca się w możliwości definiowania klas. Definicja klasy obejmuje opis stanu obiektu (dane charakteryzujące obiekt) oraz sposób zachowania (lub przetwarzania) obiektu.

Języki programowania pozwalające na programowanie obiektowe:

- Simula (1967, pierwszy język obiektowy),
- Ada,
- Object Pascal,
- Modula-3,
- C++,
- Smalltalk,
- Java.

Przykładowy program napisany techniką obiektową (Object Pascal, pakiet Delphi)

```
program FiguryGeometryczne;
  uses wincrt;

type

  Kolo = class
    x, y, promien: real;
    procedure rysuj;
  end;

  Kwadrat = class
    x, y, bok: real;
    procedure rysuj;
  end;
```

```
procedure Kolo.rysuj;
begin
  WriteLn('Rysowanie kola:');
  WriteLn('  srodek kola: (',x:0:2,', ',y:0:2,')');
  WriteLn('  promien kola: ',promien:0:2);
end;

procedure Kwadrat.rysuj;
begin
  WriteLn('Rysowanie kwadratu:');
  WriteLn('  lewy gorny naroznik: (',x:0:2,',
',y:0:2,')');
  WriteLn('  bok kwadratu: ',bok:0:2);
end;

var
  kw: Kwadrat;
  kl: Kolo;

begin
  kw := Kwadrat.Create;
  kw.x := 5;
  kw.y := 7;
  kw.bok := 4;

  kl := Kolo.Create;
  kl.x := 2;
  kl.y := 2;
  kl.promien := 1;

  kw.rysuj;
  kl.rysuj;

end.
```

Cechy języków obiektowych:

- W językach obiektowych obiekty występujące w świecie rzeczywistym reprezentowane są w postaci **klas**. Definicja klasy jest definicją **typu**. Zmienna typu klasowego nazywana jest **obiektem**.
- **Składowe** klasy: *pola (cechy)* i *metody (procedury, funkcje)*.
- **Pola** przechowują podstawowe charakterystyki obiektu rzeczywistego → **reprezentują stan obiektu rzeczywistego**.
- **Metody** to podprogramy wykonujące operacje na obiekcie → **opisują (modelują) sposób działania obiektu rzeczywistego lub sposób jego przetwarzania**.
- **Enkapsulacja** – zamknięcie w jedną całość zmiennych reprezentujących cechy obiektu oraz podprogramów opisujących sposób działania obiektu.

Sama technika programowania obiektowego wymusza wcześniejszą dogłębną analizę problemu i sensowne zaprojektowanie systemu wzajemnie powiązanych klas.

II. Wprowadzenie do programowania obiektowego w języku C++

Język C++ powstał jako rozszerzenie języka C (który sam w sobie stanowi silne narzędzie klasycznego stylu programowania) o mechanizmy wspomagające programowanie obiektowe. Główna cecha języka C++ to możliwość definiowania przez użytkownika nowych typów danych i wykorzystywania w programie obiektów zdefiniowanych typów.

Typ określa strukturę związanych z nim danych oraz definiuje operacje możliwe do wykonania na tych danych. Typ zdefiniowany poza standardowymi typami języka stanowi klasę, a obiekty tej klasy to definiowane w programie dane tego typu. Pojęciu obiektu pewnej klasy w języku C++ odpowiada pojęcie zmiennej pewnego typu w języku C.

Programowanie obiektowe to tworzenie programu przy istotnym wykorzystaniu mechanizmu definiowania nowych klas i posługiwaniu się obiektami tych klas. Sama technika programowania obiektowego wymusza wcześniejszą dogłębną analizę problemu i sensowne zaprojektowanie systemu wzajemnie powiązanych klas.

Zadanie: Rozważmy pewien system przedmiotowy tzn. system rzeczywisty, w którym występują problemy wymagające przy ich rozwiązywaniu wspomagania komputerowego. Należy napisać program, który pomoże osiągnąć rozwiązanie pewnego problemu związanego z tym systemem.

Klasyczne metody programowania:

Pojęcia i relacje występujące w rozpatrywanym systemie przedmiotowym są transponowane do zamkniętego systemu pojęć i konstrukcji języka programowania.

Programista usiłuje przy pomocy istniejących w języku standardowych typów danych i standardowych operacji zakodować wielkości obserwowane w systemie oraz jego zachowanie się.

Podczas programowania myślenie programisty biegnie dwoma odrębnymi torami:
(1) myślenie pojęciami systemu, któremu ma odpowiadać powstający program,
(2) myślenie kategoriami właściwymi językowi programowania, który stanowi narzędzie konstruowania programu.

Programowanie obiektowe:

Na podstawie analizy systemu przedmiotowego tworzy się zespół klas odzwierciedlających cechy rozważanego systemu istotne z punktu widzenia jego zachowania się. Dane składowe klasy reprezentują wielkości opisujące stan wybranej cechy systemu, a funkcje składowe przedstawiają relacje zachodzące pomiędzy elementami tego systemu.

Dalszym etapem programowania jest proces, który ma wszelkie cechy modelowania dynamiki rozpatrywanego systemu przedmiotowego. Powstały w ten sposób program uważać można za rodzaj scenariusza zachowania się tego systemu.

Podczas programowania programista myśli o programie pojęciami klas i ich obiektów, czyli pojęciami modelowanego systemu. Sam program staje się bardziej czytelny i zrozumiały nawet bez odpowiednich komentarzy.

Mechanizmy języka C++ wspomagające programowanie obiektowe

Abstrakcja danych - konstruowanie przez użytkownika własnych typów danych poprzez definiowanie odpowiednich klas. Tworzone klasy zwykle przedstawiają pewną hierarchiczną strukturę pojęć.

Hermetyzacja danych - użytkownik określa prawa dostępu do danych składowych klasy dla funkcji deklarowanych poza klasą. Pozwala to na ukrywanie danych i zapobiega ich niezamierzonemu zniszczeniu podczas komunikowania się z obiektami tej klasy.

Dziedziczenie - możliwość przeniesienia na tworzoną klasę wszystkich lub wybranych struktur danych składowych i funkcji składowych. Nowa klasa może mieć dodatkowo pewne nowe dane składowe i dodatkowe nowe funkcje składowe. Mechanizm ten pozwala w łatwy sposób budować hierarchiczną strukturę typów definiowanych przez użytkownika.

Polimorfizm - możliwość odwoływania się do funkcji (tzw. funkcji wirtualnych), które jeszcze nie zostały zdefiniowane w danej klasie stanowiącej bazę dziedziczenia.

III.Standardowe wejście - wyjście strumieniowe

Wymiana danych z urządzeniami wejścia - wyjścia w języku C++ może być realizowana poprzez wykorzystanie:

funkcji bibliotecznych języka C (np. getchar, getch, putchar, scanf, printf) przeznaczonych do wprowadzania i wyprowadzania danych typów wbudowanych języka (char, int, float, double),

strumieniowych mechanizmów języka C++, które umożliwiają zarówno wprowadzanie danych typów wbudowanych, jak i zaprogramowanie operacji wejścia - wyjścia dla danych typów definiowanych przez użytkownika.

Strumieniowe wprowadzanie i wyprowadzanie danych w języku C++ związane jest z hierarchicznym układem klas strumieniowych zdefiniowanych w bibliotekach większości kompilatorów języka. Główne klasy strumieniowe to **`istream`** (wprowadzanie danych) i **`ostream`** (wyprowadzanie danych) oraz **`iostream`** (klasa dziedzicząca wielobazowo istream i ostream). Każda zdefiniowana klasa może służyć tworzeniu obiektów strumieniowych, które pośredniczą wprowadzaniu i wyprowadzaniu danych.

Dla strumieni standardowych zostały (zwykle w pliku nagłówkowym **`iostream.h`**) predefiniowane obiekty:

`cin` - standardowy strumień wejściowy (związany z klawiaturą),
`cout` - standardowy strumień wyjściowy (związany z monitorem),
`cerr` - standardowy strumień diagnostyczny (związany z monitorem).

Operatory >> i <<

Wprowadzając lub wyprowadzając dane w sposób obiektowy wykorzystywać można przeciążone dwuargumentowe operatory >> i <<. Lewym argumentem każdego z operatorów musi być obiekt strumieniowy, z którego lub do którego dane mają być przesyłane (**`cin`** dla operatora >> oraz **`cout`** lub **`cerr`** dla operatora <<). Prawy argument opisuje wprowadzane lub wyprowadzane obiekty w pamięci operacyjnej:

Operator **>> wprowadza** dane ze strumienia **`cin`** i umieszcza je w zmiennej będącej prawym argumentem.

Operator **<< wyprowadza** wartość wyrażenia będącego prawym argumentem do strumienia **`cout`**.

Wprowadź I uruchom program, ZINTERPRETUJ KOD ŹRÓDŁOWY:

```
#include <iostream>
using namespace std;
int main()
{
   char c;
   int i;
   float f;
   cin >> c;
   cin >> i;
   cin >> f;
   cout << c;
   cout << i;
   cout << f;
}
```

K	**KLAWIATURA**	K↵
123	**KLAWIATURA**	123↵
9.876543	**KLAWIATURA**	9.876543↵
K1239.876543		

Wartością wyrażenia **`cin >> zmienna`** jest *referencja* do obiektu **`cin`**. Pozwala to na konstruowanie łańcuchów (sprzęganie) operatorów >>. Operatory >> i << mają wiązanie lewostronne, czyli wartość wyrażenia:

```
cin >> k >> n
```

jest wyliczana jako:

```
(cin >> k) >> n
```

Efektem sprzężenia operatorów >> jest sukcesywne wprowadzanie danych ze strumienia **`cin`** do kolejnych zmiennych występujących od lewej strony w wyrażeniu strumieniowym. Tak samo tworzyć można łańcuchy operatorów **`<<`** wyprowadzające do strumienia **`cout`** wartości wielu wyrażeń.

Sprzęganie operatorów

Wprowadź I uruchom program, ZINTERPRETUJ KOD ŹRÓDŁOWY:

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

int main()
{
   char c;
   int i;
   float f;
   cin >> c >> i >> f;
   cout << c << i << f;
}
```

Priorytet operatorów >> i <<

Operatory strumieniowe >> i << powstały przez przeciążenie standardowych operatorów bitowego przesunięcia w prawo i w lewo. W związku z tym operatory strumieniowe zachowują standardowy priorytet operatorów przesunięcia. W niektórych przypadkach konieczne staje się użycie nawiasów w wyprowadzanych wyrażeniach.

Uzasadnij komentarze w poniższych przypadkach:

```
int a=12,b=3;                    // Wyświetlane:
cout << a + b;                   // Jedna liczba 15
cout << (a < b);                 // Jedna liczba 0
cout << (a > b ? a : b);         // Jedna liczba 12
cout << a << b;                  // Dwie liczby 12 3
cout << (a << b);                // Jedna liczba 96
cout << (a = b);                 // Jedna liczba 3
```

Formatowanie wejścia - wyjścia strumieniowego

Operatory strumieniowe nie pozwalają bezpośrednio na formatowanie danych na wzór klasycznych funkcji języka C (np. scanf, printf). Redagowania danych można dokonać poprzez użycie **manipulatorów** lub jawne wprowadzanie do strumienia odpowiednich łańcuchów znakowych.

Wprowadź I uruchom program, ZINTERPRETUJ KOD ŹRÓDŁOWY:

```
#include <iostream>
#include <stdio.h>
using namespace std;

int main()
{
   int a=12,b=3;
   printf("\nSuma %d plus %d wynosi %d",a,b,a+b);
   cout <<"\nSuma "<< a <<" plus "<< b <<" wynosi "<< a+b;
}
```

```
Suma 12 plus 3 wynosi 15
Suma 12 plus 3 wynosi 15
```

Standardowa współpraca ze strumieniem:

- Obiekty typu **`char`** są związane z wprowadzaniem lub wyprowadzaniem **znaków** ASCII.
- Obiekty typu **`char*`** powodują wprowadzanie lub wyprowadzanie **łańcuchów znakowych**. Wprowadzanie łańcucha kończy pojawienie się w strumieniu wejściowym **odstępu**.
- Obiekty typu arytmetycznego (**`int, long int, float, double, long double`**) biorą udział przy wprowadzaniu lub wyprowadzaniu **liczb**.
- Liczby całkowite są wyprowadzane dziesiętnie, a wprowadzane dziesiętnie, ósemkowo lub szesnastkowo zgodnie z ich zapisem w strumieniu **`cin`** (*cc...c, 0cc...c, 0xcc...c*).
- Liczby rzeczywiste są wyprowadzane ze standardową dokładnością **6** cyfr ułamkowych, a do wprowadzania może być użyta notacja zwykła lub naukowa.
- Podczas wprowadzania danych ze strumienia **`cin`** następuje pomijanie w strumieniu wejściowym wszystkich początkowych odstępów. Odstępy takie są pomijane również w przypadku wprowadzania znaku do zmiennej typu **`char`**.

Manipulatory

W strumieniowym wprowadzaniu i wyprowadzaniu danych do ich redagowania wykorzystuje się pewien rodzaj funkcji zwanych manipulatorami. Manipulatory mogą być bezparametrowe lub zawierać argumenty. Najważniejszą cechą manipulatora jest sposób jego wywołania poprzez umieszczenie go w miejscu prawego argumentu operatora >> lub <<. Wynikiem wykonania operatora strumieniowego z manipulatorem jest referencja do strumienia **`cin`** lub **`cout`**, co pozwala włączać manipulatory do dłuższych łańcuchów tych operatorów.

Manipulatory bezparametrowe:

- **endl** - wyprowadzenie nowej linii,
- **dec** - dziesiętna konwersja liczb całkowitych,
- **hex** - szesnastkowa konwersja liczb całkowitych,
- **oct** - ósemkowa konwersja liczb całkowitych,
- **ws** - pominięcie odstępów przy wprowadzaniu danych.

Wprowadź I uruchom program, ZINTERPRETUJ KOD ŹRÓDŁOWY:

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

int main()
{
  int a,b,c;
  cin >> hex >> a >> b >> c;
  cout << hex << a << endl << dec << b << endl << oct <<c;
}
```

```
2F FF D1          KLAWIATURA      2F FF D1↵
2f
255
321
```

Manipulatory z parametrami:

`setw(n)` - ustalenie szerokości pola w strumieniu wejściowym lub wyjściowym na **n** znaków (wpływa jedynie na najbliższą operację wprowadzania),

`setprecision(n)` - ustalenie na **n** liczby cyfr części ułamkowej liczby rzeczywistej,

`setfill(n)` - ustalenie znaku o kodzie **n** do wypełniania nadmiarowej szerokości pola wyjściowego

Wprowadź i uruchom program, ZINTERPRETUJ KOD ŹRÓDŁOWY:

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

int main()
{
  int a=1234, b=775;
  cout<<setw(10)<<setfill('*')<<a<<","<<setw(8)<<
    setfill('#')<<b;
}
```

```
******1234,#####775
```

Zadanie CPP01: (zadanie należy posiadać na pendrive'ie na kolejnych zajęciach!)

Napisz program **z użyciem strumieniowych mechanizmów języka C++ (cin, cout),** który wczytuje z klawiatury 3 liczby całkowite (o wartości mieszczącej się w typie **long int**), po czym wypisuje je **w kolumnie** z zachowaniem **pionowego wyrównania** poszczególnych pozycji dziesiętnych.

Funkcje składowe klas strumieniowych

W klasach strumieniowych zdefiniowane zostały funkcje składowe wspomagające wprowadzanie i wyprowadzanie danych. Funkcje te wywoływane są na rzecz właściwego obiektu (strumienia) zgodnie ze standardowym sposobem wywołania funkcji składowej klasy:

nazwa_strumienia . nazwa_funkcji

`get(char&)` - funkcja pobiera jeden znak ze strumienia **`cin`** i umieszcza jego kod w argumencie typu referencja do **`char`**. Zwracaną przez funkcję wartością jest referencja do strumienia (lub NULL w przypadku wystąpienia końca pliku), co pozwala tworzyć łańcuchy wywołań tej funkcji. Funkcja wprowadza każdy znak ze strumienia, nawet znak odstępu (inaczej niż operator **>>**).

`put(char)` - funkcja wyprowadza do strumienia **`cout`** znak podany jako argument. Zwracaną wartością jest referencja do strumienia. Umożliwia to tworzenie łańcucha wywołań funkcji.

Wprowadź I uruchom program, ZINTERPRETUJ KOD ŹRÓDŁOWY:

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

int main()
{
   char a,b,c,d,e;
   cin.get(a).get(b).get(c).get(d).get(e);
   cout.put(e).put(d).put(c).put(b).put(a);
}
```

```
Zebra                KLAWIATURA      Zebra↵
arbeZ
```