Lecture 1 examples

October 7, 2019

1 From C to C++ with (out) agonizing pain

Postaramy si przedstawi pewne nowe elementy jezyka, za którymi sta ma to cae ++
Na razie bez obiektowoci

1.1 Prosty program

Bdziemy uywa fragmentów kodu w celu pokazania rónic. Ponisze fragmty nie musz si kompilowa, a maja jedynie suyc ilustracji pewnych konceptów.

Wczytaj warto i wydrukuj na ekran * implementacja w C

```
In [ ]: // Podstawowy program w jzyku C
        #include <stdio.h>
        int main ()
        {
            int a;
            scanf("%d", &a);
            printf ("Hello!a=%d\n", a );
        }
```

Wykonywalny fragment kodu:

```
In [1]: #include <stdio.h>

Out[1]:

In [2]: int a;
        a = 10;
        printf ( " Hello! a =%d\n", a );

 Hello! a =10


Out[2]: (int) 14
```

- Implementacja w C++ Pojawiaj si nowe elementy:
- std::cin zamiast scanf

- std::cout zamias printf
  std oznacza standartow przestrze nazw (patrz niej)

In [ ]:
```
// Ten sam program w C++
#include <iostream>

int main()
{
    int a;
    std::cin >> a ;
    std::cout << "Hello! a=" << a << std::endl ;
}
```

In [1]:
```
#include <iostream>
```

Out[1]:

In [4]:
```
a = 5;
std::cout << "Hello! a=" << a << std::endl ;
```

```
Hello! a=5
```

Out[4]:
```
(std::basic_ostream<char, std::char_traits<char> >::__ostream_type &) @0x7f69b3725e60
```

1.2 Przestrzenie nazw - namespace

Pierwzym nowym elementem s przestrzenie nazw (namspace) pozwalajce na uporzdkowanie kodu.

In [2]:
```
namespace A // Deklaracja przesrzeni nazw
{
    int fun(); // Deklaracja funkcji wewntrz przestrzeni
}
namespace B
{
    namespace C
    {
        namespace D // Mona zagniedzic przestrzenie nazw
        {
            int fun() {return 3;} //Definiujemy fun w B::C::D
        }
    }
}
```

Out[2]:

Zdefiniujemy zadeklarowan w przestrzeni **A** funkcj *int fun()*. Dostp do przestrzeni odbywa si przez operator zasigu ::

```
In [3]: //Definiujemy fun() z A
        int A::fun()
        {
          return 1;
        }

Out[3]:

In [8]: std::cout << "Wywoanie fun() z A \t\t" << A::fun() << std::endl;
        std::cout << "Wywoanie fun() z B::C::D \t" <<B::C::D::fun() << std::endl;

Wywoanie fun() z A                1
Wywoanie fun() z B::C::D        3


Out[8]: (std::basic_ostream<char, std::char_traits<char> >::__ostream_type &) @0x7fe18013ae60
```

Mona zadeklarowa uzywanie konkretnej przestrzeni nazw w kodzie uywajc ***using namespace nazwa_przestrzeni***

```
In [9]: using namespace B;
        std::cout << C::D::fun() << std::endl;

3


Out[9]: (std::basic_ostream<char, std::char_traits<char> >::__ostream_type &) @0x7fe18013ae60
```

Albo kilka: ***using namespace nazwa_przestrzeni1::nazwa_przestrzeni2***

```
In [10]: using namespace B::C::D;
         std::cout << C::D::fun() << std::endl;
         std::cout << A::fun() << std::endl;

3
1


Out[10]: (std::basic_ostream<char, std::char_traits<char> >::__ostream_type &) @0x7fe18013ae60
```

Naley by jednak ostronym. Co oznacza bdzie nastpujcy fragment:

```
In [11]: using namespace A;
         using namespace B::C::D;
         std::cout << fun() << std::endl;
```

```
input_line_14:4:14: error: call to 'fun' is ambiguous
std::cout << fun() << std::endl;
             ^~~input_line_4:11:17: note: candidate function
            int fun() {return 3;} //Definiujemy fun w B::C::D
                ^input_line_5:2:8: note: candidate function
int A::fun()
       ^
```

Widzimy, e nasz kod spowodowa bd.

1.3 Nowe elementy

1.3.1 vector

Dynamiczna kolekcja zdefiniowana w bibliotece standartowej, o kótrej powiemy wicej póniej. Pozwala przechowywa w sposób cigy elementy. Przykad:

```
In [ ]: #include <iostream>
        #include <vector> //definicja vector

        using namespace std; // okrelenie przestreni nazw

        int main()
        {
            vector<vector<int> > a(5); // vector of vectors - tablica
            for(int i=0; i<a.size(); ++i)// dostosuj rozmiar
                a[i].resize(5);

           a[0][1] = 5; // inicjalizacja
        }
```

1.3.2 New and delete

Troch bardziej przyjazne zarzdzanie zasobami. Poniekd zastpuje malloc() i free().
W poniszym przykadzie stworzymy struktur D zawierajc dwie wielkoci typu int.

```
In [14]: struct D{
             int a;
             int b;
         };
```

```
Out[14]:
```

Uzyjemy **new** by stworzy instancje. **new** kontroluje typ zwracanej zmiennej, przez co nie musimy rzutowa wyniku na porzdany typ, co byo konieczna w przypadku malloc().

```
In [15]: int n=10;
         D * p = new D;
         D * tab = new D[n];
```

```
Out[15]:

In [16]: tab[8].a = 1;
         tab[8].b = 2;

Out[16]: (int) 2


In [17]: using namespace std;
         cout << tab[8].a << endl;

1


Out[17]: (std::basic_ostream<char, std::char_traits<char> >::__ostream_type &) @0x7fe18013ae60


In [18]: delete p;
         delete []tab;

Out[18]: (void) nullptr
```

1.4 Referencja

Pozwala na nadanie zmiennej aliasu i wygodniejsze przekazywanie jej do funcji.

Najpier jak to byo w C?

```
In [1]: #include <iostream>

Out[1]:

In [20]: void fun1(int a){
             a = 5;
         }

Out[20]:
```

Zmienna przekazana przez warto. Operacja wykonana na kpi.

```
In [21]: int b = 10;
         fun1(b);
         std::cout << b << std::endl;

10


Out[21]: (std::basic_ostream<char, std::char_traits<char> >::__ostream_type &) @0x7fe18013ae60
```

Moe z urzyciem adresu?

```
In [22]: void fun2(int* a){
             *a = 5;
         }

Out[22]:

In [23]: fun2(&b);
         std::cout << b << std::endl;

5


Out[23]: (std::basic_ostream<char, std::char_traits<char> >::__ostream_type &) @0x7fe18013ae60
```

C++ pozwala, na uycie referencji poprzez uycie **&**

```
In [24]: // Funkcja z referencj
         void fun3(int &a){
             // tu wydrukuj adres zmiennej
             a = 5;
         }

Out[24]:

In [25]: int c = 10;
         fun3(c);
         std::cout << c << std::endl;

5


Out[25]: (std::basic_ostream<char, std::char_traits<char> >::__ostream_type &) @0x7fe18013ae60
```

1.4.1 const zamiast define

```
In [26]: #define PI 3.141592
         const double pi = 3.141592;

Out[26]:

In [27]: std::cout << pi << std::endl;

3.14159


Out[27]: (std::basic_ostream<char, std::char_traits<char> >::__ostream_type &) @0x7fe18013ae60
```

1.5 Struktura z C

W jzyku C moliwe byo wykorzystanie struktur w celu pogrupowania danych.

```
In [1]: struct A{
            int a;
            int b;
        };

Out[1]:

In [2]: A a1;

Out[2]:

In [3]: A *p = &a1;

Out[3]:

In [4]: a1.a = 12;
        p->b = 15;

Out[4]: (int) 15


In [6]: std::cout << a1.a << " " << a1.b << std::endl;

12 15


Out[6]: (std::basic_ostream<char, std::char_traits<char> >::__ostream_type &) @0x7f5b3c169e60


In [7]: std::cout << p->a << " " << p->b << std::endl;

12 15


Out[7]: (std::basic_ostream<char, std::char_traits<char> >::__ostream_type &) @0x7f5b3c169e60
```

Jzyk C++ wprowadza obok struktur dodatkowo pojcie klasy. Elementem rónicym sa modyfikatory dostpu (public, privite i protected) pozwalajce na osignicie hermetyzacji. Domylnie wszystko wewntrz klasy jest prywatne, czyli nie do dotknicia z zewntrz:

```
In [2]: class B{
            int a;
            int b;
        };

Out[2]:
```

```
In [3]: B b1;
        B *p=&b1;
```

```
Out[3]:
```

Nie moemy dosta si do czonków B

```
In [5]: b1.a = 12;
```

```
input_line_7:2:5: error: 'a' is a private member of 'B'
 b1.a = 12;
    ^input_line_4:2:9: note: implicitly declared private here
    int a;
        ^
```

```
In [6]: p->b = 10;
```

```
input_line_8:2:5: error: 'b' is a private member of 'B'
 p->b = 10;
    ^input_line_4:3:9: note: implicitly declared private here
    int b;
        ^
```

Zmiemy dostpnoc przy uyciu modyfikatora public:

```
In [2]: class B{
            public:
            int a;
            int b;
        };
```

```
Out[2]:
```

```
In [4]: B b1;
        B *p=&b1;
        b1.a = 12;
        p->b = 10;
```

```
Out[4]: (int) 10
```

```
In [5]: std::cout << p->a << " " << p->b << std::endl;
```

```
12 10
```

```
Out[5]: (std::basic_ostream<char, std::char_traits<char> >::__ostream_type &) @0x7fbcb5115e60
```

Teraz moemy zarówno modyfikowa, jak i czyta wartoci zmiennych zadeklarowanych wewntrz klasy B