Język C++ zajęcia nr 2

Inicjalizacja

Definiowanie obiektu może być połączone z nadaniem mu wartości początkowej za pomocą inicjalizatora, który umieszczany jest po deklaratorze obiektu. W języku C++ inicjalizator może mieć formę:

= wyrażenie;

= { lista inicjalizatorów };

(lista wyrażeń);

Wyrażenia występujące w inicjalizatorze mogą być dowolnymi poprawnymi wyrażeniami języka C++.

Inicjalizator w formie listy wyrażeń w nawiasach jest używany między innymi do wywołania odpowiedniego konstruktora dla definiowanego obiektu klasy. Może być również stosowany do inicjalizowania nieobiektowych typów danych.

```
int a(16);          // a=16

int b(a+4);         // b=20
```

Definicja zmiennej w instrukcji `for`

W instrukcji **for** w miejscu pierwszego wyrażenia może wystąpić **definicja** wprowadzająca zmienne wykorzystywane w pętli:

`for (` ***def wyr2*** **;** ***wyr3*** **)** ***inst***

Definicja ***def*** może być połączona z inicjalizacją definiowanych zmiennych.

Wprowadź i uruchom program:

```
#include <stdio.h>
using namespace std;
int main()
{
   for(long s=1,k=1;k-13;s*=++k)printf("%2ld! = %ld\n",k,s);
}
```

```
 1! = 1
 2! = 2
 3! = 6
 4! = 24
 5! = 120
 6! = 720
 7! = 5040
 8! = 40320
 9! = 362880
10! = 3628800
11! = 39916800
12! = 479001600
```

Referencje

Pojęcie referencji jest ściśle związane z przekazywaniem argumentów przy wywołaniu funkcji i zwracaniem przez nią wyniku.

W języku **C** argumenty mogą być przekazywane jedynie przez **wartość**, co oznacza, że tworzona jest kopia aktualnej wartości argumentu, która po przekazaniu do funkcji, odbierana jest tam jako parametr formalny. W funkcji można zmienić wartość zmiennej będącej parametrem formalnym, co jednak nie zmienia oryginalnej wartości argumentu.

W języku **C++** argumenty mogą być przekazywane **zarówno przez wartość, jak i przez referencję.**

Referencja do pewnej zmiennej jest inną nazwą, synonimem tej zmiennej. Obowiązuje zasada:

> **Każda operacja wykonana na zmiennej referencyjnej w rzeczywistości wykonana zostanie na zmiennej, do której odnosi się referencja**.

W deklaracji, **nazwa zmiennej referencyjnej** poprzedzona jest symbolem **&**.

Deklarowana zmienna referencyjna **musi być inicjalizowana**. Inicjalizatorem jest najczęściej **l-wartość** (zwykle nazwa zmiennej), **do której odnosi się deklarowana referencja**. Inicjalizator musi mieć w takim przypadku typ zgodny z typem zmiennej referencyjnej.

Uwaga: Można deklarować zmienne będące referencjami do różnych typów, jednak bez tworzenia **referencji do referencji**, **tablic referencji** oraz **wskaźników na referencje.**

Wprowadź i uruchom program, zinterpretuj wyniki:

```
#include <iostream>
using namespace std;
int main()
{
   int k=16;                          // k otrzymuje wartość 16
   int &rk=k;                         // rk jest referencją do k
   cout << k << " " << rk << endl;
   rk=8;                              // faktycznie następuje zmiana k
   cout << k << " " << rk << endl;
}
```

```
16 16

8 8
```

Po dokonaniu inicjalizacji zmiennej referencyjnej nie można zmienić jej referencji, co oznacza, że **referencja stale odnosi się do tego samego obiektu**.

Uwaga: W zapisie deklaracji referencji nie musi występować spacja. Umieszcza się ją jedynie dla podniesienia czytelności tekstu programu.

Równoważne zapisy deklaracji:

```
int & p=q;

int &p=q;

int& p=q;

int&p=q;
```

Każdy operator działający na zmiennej referencyjnej (zgodnie z ogólną zasadą dotyczącą referencji) w rzeczywistości działa na zmiennej, z którą dana referencja jest związana. W szczególności operator adresu zastosowany do zmiennej referencyjnej daje adres zmiennej, z którą dana zmienna referencyjna jest związana (np. jeżeli **`int &y=x;`** to wyrażenie **`&y`** zwraca wartość **`&x`**, czyli adres zmiennej **`x`**)

Wprowadź i uruchom programy, zinterpretuj wyniki:

Adres zmiennej referencyjnej:

```
#include <stdio.h>
using namespace std;
int main()
{
   int x;
   int &y=x;
   printf("%p\n%p",&x,&y);          // Wyświetlenie adresów zmiennych
}
```

```
0028FF44

0028FF44
```

(rezultaty mogą być różne na różnych maszynach)

Referencje do tablic. Indeksowanie zmiennej referencyjnej:

```
#include <iostream>
using namespace std;
int main()
{
   int a[8]={0,1,2,3,4,5,6,7};
   int (&b)[8]=a;              // b jest referencją do tablicy a
                               // int &b[8] byłoby tablicą referencji

   cout <<"a[5]="<<a[5]<<"  b[5]="<<b[5]<<endl;
}
```

```
a[5]=5  b[5]=5
```

Referencje do stałych i zmiennych

Zmienna referencja do zmiennej ma miejsce, gdy zmienna referencyjna i jej inicjalizator nie są deklarowane z modyfikatorem **`const`** oraz inicjalizator nie jest literałem lub stałym wskaźnikiem. **Referencja sprawia, że każda modyfikacja zmiennej referencyjnej powoduje identyczną modyfikację zmiennej, która była inicjalizatorem. Również każda zmiana zmiennej inicjalizacyjnej pociąga za sobą taką samą zmianę zmiennej referencyjnej.**

Referencja do zmiennej (zinterpretuj program)**:**

```
#include <iostream>
using namespace std;
int main()
{
   int a=4;
   int &b=a;
   cout <<"a="<<a<<" b="<<b<<endl;
   a++;
   cout <<"a="<<a<<" b="<<b<<endl;
   b++;
   cout <<"a="<<a<<" b="<<b<<endl;
}
```

```
a=4 b=4

a=5 b=5

a=6 b=6
```

Użycie modyfikatora **`const`** w deklaracji zmiennej referencyjnej powoduje, że zmienna referencyjna **nie może być później bezpośrednio modyfikowana**. Inicjalizatorem deklarowanej w taki sposób zmiennej referencyjnej **nie musi być l-wartość** oraz nie jest wymagana zgodność typu inicjalizatora z typem zmiennej referencyjnej.

Stała referencja do zmiennej dotyczy zmiennej referencyjnej deklarowanej z modyfikatorem **const** i inicjalizowanej **l-wartością**. Nie można modyfikować samej zmiennej referencyjnej bezpośrednio, **ale można zmieniać jej inicjalizator**.

Stała referencja do zmiennej (zinterpretuj program)**:**

```
#include <iostream>
using namespace std;
int main()
{
   int a=4;
   const int &b=a;
   cout <<"a="<<a<<" b="<<b<<endl;
   a++;
   cout <<"a="<<a<<" b="<<b<<endl;
   // b++;   Błąd, nie można modyfikować stałej
}
```

```
a=4 b=4

a=5 b=5
```

Stała referencja do stałej występuje, gdy zmienna referencyjna i jej inicjalizator są deklarowane z modyfikatorem **const**. W takim przypadku nie jest możliwa zmiana wartości zmiennej referencyjnej ani jej inicjalizatora.

Stała referencja do stałej (zinterpretuj program)**:**

```
#include <iostream>
using namespace std;
int main()
{
   const int a=4;
   const int &b=a;
   cout <<"a="<<a<<" b="<<b<<endl;
   // a++;   Błąd, nie można modyfikować stałej
   // b++;   Błąd, nie można modyfikować stałej
}
```

```
a=4 b=4
```

Główne wykorzystanie referencji wiąże się z przekazaniem argumentów do wywołanej funkcji. Przekazanie argumentu przez referencję wymaga umieszczenia w deklaracji funkcji odpowiedniego parametru formalnego typu referencyjnego. W momencie wywołania funkcji przekazanie argumentu do parametru typu referencyjnego odbywa się w sposób, którego konsekwencje są identyczne do inicjalizacji zmiennej referencyjnej.

Wprowadź i uruchom program, zinterpretuj wyniki:

Referencyjne przekazanie argumentu:

```
#include <iostream>
using namespace std;
void fun(int a, int &b)
      // Każde wywołanie inicjalizuje referencyjny parametr b
{
   a++;                          // Inkrementacja zmiennej a
   b++;                          // Faktycznie inkrementacja zmiennej y
   cout << "               Wewnatrz funkcji a=" << a
        << " b=" << b << endl;
   return;
}
int main()
{
   int x=6,y=8;
   cout << "       Przed wywolaniem funkcji x=" << x
        << " y=" << y << endl;
   fun(x,y);
   cout << "Po powrocie z wywolania funkcji x=" << x
        << " y=" << y << endl;
}
```

```
       Przed wywolaniem funkcji x=6 y=8

               Wewnatrz funkcji a=7 b=9

Po powrocie z wywolania funkcji x=6 y=9
```

Mechanizm referencji kompilator języka C++ zwykle realizuje poprzez przekazywanie adresu obiektu. Daje to dodatkową korzyść implementacyjną:

- Oszczędność czasu - nie trzeba wykonywać dodatkowych rozkazów układających na stosie struktury danych przekazywane jako argumenty,
- Oszczędność pamięci - mniejsze zapotrzebowanie na przestrzeń stosu.

Również **zwracany przez funkcję rezultat może być typu referencyjnego**. Daje to taki efekt, jakby w tekście programu w miejscu wywołania funkcji po powrocie z niej znajdowała się pewna **l-wartość**. Wynikają z tego wszystkie konsekwencje związane z l-wartością, w szczególności wywołanie takiej funkcji może wystąpić jako modyfikowany argument operatorów przypisania lub inkrementacji.

Wprowadź i uruchom program, zinterpretuj wyniki:

Referencja zwracana przez funkcję:

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

int kot=4;              // Zmienna globalna

int& kotek()            // Referencyjny rezultat funkcji
{
   return kot;          // Zwracanym rezultatem jest referencja do kot
}

int main()
{
   cout << "kot=" << kot << endl;
   kotek()=7;
   cout << "kot=" << kot << endl;
   kotek()++;
   cout << "kot=" << kot << endl;
}
```

```
kot=4

kot=7

kot=8
```

Uwaga: W przypadku funkcji zwracających referencję l-wartość występująca w instrukcji `return` musi reprezentować obiekt istniejący po powrocie z wywołania funkcji (nie może to być np. obiekt lokalny zdefiniowany wewnątrz tej funkcji).

Zadanie dla zainteresowanych (opcjonalne) – zaliczenie możliwe wyłącznie na następnych zajęciach po przedstawieniu wydruku (kod źródłowy, okno z efektami uruchomienia) i po ustnym zaliczeniu.

Napisz i uruchom program, który na szachownicy (8 x 8) ustawia losowo *n* pionków (wartość *n* podaje użytkownik, *n*<=6000, na każdym polu może stanąć wiele - max. 99 - pionków), po czym drukuje tablicę z liczbą pionków znajdujących się na każdym polu. Wymagania dodatkowe:

- **wyznaczenie współrzędnych pionka** (każda współrzędna = 0, 1, 2, 3, 4, 5, 6, *lub* 7) wykonywane jest przez **osobną funkcję** *ustaw* (wywoływaną z funkcji *main*),

- **przekazywanie argumentów** (obie współrzędne położenia pionka) pomiędzy funkcją *main* i *ustaw* **musi odbywać się przez referencje**: *void ustaw(int &x, int &y)*

- wybór każdego pola planszy musi być w przybliżeniu **jednakowo prawdopodobny**, *wskazówka: wyznacz współrzędną jako resztę z dzielenia pseudolosowej liczby całkowitej (wygenerowanej przeze rand()) przez 8,*

- do operacji wejścia-wyjścia **użyj mechanizmów strumieniowych** (cin, cout).

- w funkcji *main* zadeklaruj dwuwymiarową tablicę (tablicę tablic) 8 x 8 (np. *int szachownica[8][8]*), wczytaj (z użyciem cin) wartość *n*, zastosuj pętlę realizującą *n* wywołań funkcji *ustaw* (modyfikując po każdym wywołaniu odpowiedni element tablicy), a na końcu wydrukuj tablicę z pionowym wyrównaniem pozycji dziesiętnych w poszczególnych kolumnach, np.:

```
 0    1    0    6    5    7    2    1
 1   12    5    0   14   22   56    3
12    7   …
…
```