Jakub Łabendowicz 08.05.2020r.

Sprawozdanie

LABORATORIUM 9.

# Pytania kontrolne:

## Na czym polega polimorfizm statyczny?

Określa on jakie metody będą wykonywane już w momencie kompilacji programu.

## Na czym polega polimorfizm dynamiczny?

Określa on jakie wersje metody(virtual) będą wykonywane dopiero w czasie wykonywania programu.

## Czy w języku C++ jest możliwe połączenie polimorfizmu dynamicznego i statycznego?

Tak, jest możliwe używania szablonów klas template<>

## Czy mechanizm wartości domyślnych argumentów metod zalicza się do polimorfizmu?

Nie.

## Czy polimorfizm statyczny się dziedziczy?

Może być dziedziczony.

## Czy argumenty domyślne się dziedziczy?

Tak, jeżeli w klasie są zdefiniowane wartości domyślne dla parametrów metod i dana klasa jest dziedziczona.

## Co się dzieje, gdy istnieje już polimorfizm, ale usunie się słowo virtual?

Wykonane byłyby metody zgodne z typem wskaźnika lub referencji.

## Po co są te „polimorfizmy”?

Dzięki niemu mamy kontrolę nad wykonywaniem programu, w momencie kompilacji i podczas działania programu – niezależnie od wyborów użytkownika.

## W jaki sposób w języku C++ pobiera się i zwalnia pamięć?

Do alokowania pamięci służy operator new. Do zwalniania operator delete.

# Zadanie 9.1. Zbadanie rozmiarów wybranych typów wskaźnikowych

## Czy rozmiar zmiennej wskaźnikowej zależy od typu na jaki ten wskaźnik pokazuje?

Nie, nie jest to zależne od typu danych.

# Zadanie 9.2. Alokacja pamięci dla zmiennej skalarnej

## Listing 9.2, wiersz 11 → Jaki byłby efekt, gdyby tego wiersza nie było?

Alokujemy pamięć korzystając z operatora new, więc należy ją zawsze usuwać, wtedy kiedy wskaźnik nie jest już nam potrzebny, w ten sposób nie zużywamy bez sensu pamięci operacyjnej

## Listing 9.2, wiersze 5 i 7 → Po co jest znak\*?

Odwołujemy się do wskaźnika.

## Co oznacza zapis: int \*\* wskInt2D = nullptr?

Jest to deklaracja zmiennej wskaźnikowej do wskaźnika, który pokazuje na typ zmiennej int z wartością początkową nullptr, czyli pustym adresem.

# Zadanie 9.3. Alokacja pamięci dla tablicy jednowymiarowej

## Listing 9.3, wiersz 4 → Ile pamięci zostanie zalokowane ?

3 bajty

## Listing 9.3, wiersze 7 i 11 → Co powoduje operator + ?

\*(wskInt1D+2) jest to inny sposób na odwołanie do wskaźnika, które jest równoważne zapisowi wskInt1D[2].

## Listing 9.3, wiersze 15 i 17 → Co powoduje operator ++ ?

++ przechodzi do następnego bajta pamięci

# Zadanie 9.4. Alokacja pamięci dla tablicy dwuwymiarowej

## Dlaczego lokowanie pamięci odbywa się dwuetapowo?

Lokujemy pamięć do dwóch zmiennych. Tworzymy wskaźnik do zmiennej odpowiadającej za wiersze tablicy, następnie dla każdej wartości w tworzymy nowy wskaźnik do zmiennej odpowiadającej za kolumny tablicy dwuwymiarowej.

## Dlaczego zwalnianie pamięci odbywa się dwuetapowo?

Najpierw trzeba zwolnić komórki pamięci na które pokazują wskaźniki do kolumn tablicy dwuwymiarowej, następnie usuwamy wskaźnik do wierszy tablicy.

# Zadanie 9.5. Referencje a wskaźniki

## Listing 9.5, wiersze 5,6 → Dlaczego to nie zadziała?

Odwołujemy się do adresu pustej komórki pamięci. wykonując operacje w taki sposób zmieniamy adres tej zmiennej, a nie zawartość tej zmiennej.

## Listing 9.5, wiersz 14 → Co się tutaj dzieje?

Do referencji przypisaliśmy adres zmiennej a, więc kiedy przypiszemy do referencji wartość zmiennej b, to wartość zmiennej a przyjmie wartość zmiennej b, gdyż do referencji adres można przypisać tylko raz.

## Listing 9.5, wiersze 14 i 24 → Jaka jest różnica?

Do referencji adres można przypisać tylko raz, wskaźniki można modyfikować.

## Czy rozmiar referencji zależy od typu na jaki ten wskaźnik pokazuje? (Czy takie pytanie ma w ogóle sens?)

Nie zależy.