Sam dla siebie C++
każdy programować może

piątek, dnia 24.10. 2025 r.

Wskaźniki w C++ nr 009/25

Wskaźniki.

Wskaźniki, jeden termin w słowniku a wiele znaczeń. Prawda? Język polski piękny jest. I jakby tu nie ująć, obojętnie jaki to przedmiot zostanie użyty do pokazywania czegoś, tytułuje się go nie inaczej tylko – wskaźnik. No cóż, tak to bywa. Wskaźnikiem może być zakładka w książce, notatka zawierające potrzebną informację zapisana na malutkim skrawku papierku. A także – palec użyty w formie zakładki w czasie szukania informacji w książce. W takim razie - wskaźnik co to takiego jest?

A zatem, cytat:

Wskaźnik - to zmienna, która mierzy lub odzwierciedla stan badanej cechy lub zjawiska, np. poprzez łatwe do zaobserwowania fakty, zdarzenia czy zjawiska. W kontekście naukowym i technicznym jest to zazwyczaj liczba wyrażająca stosunek dwóch wielkości lub dane z analizy. Może również odnosić się do fizycznego przyrządu (np. wskaźnika na zegarze) lub w informatyce do typu danych wskazującego na adres pamięci.

W informatyce:

Wskaźnik (ang. *pointer*) to zmienna, która przechowuje adres pamięci innej zmiennej. Służy do pośredniego dostępu do danych w celu np. efektywnego zarządzania pamięcią.

W programowaniu, zwłaszcza w językach takich jak C++, wskaźnik to zmienna, która przechowuje adres innej zmiennej. Dzięki temu możliwe jest bezpośrednie manipulowanie pamięcią komputera.

I o tym wskaźniku właśnie będzie mowa, bo do mojego hobbistycznego gromadzenia i praktykowania języka C++ , taka wiedza była i jest potrzebna.

Informuję, że moja wiedza oparta jest ściśle na materiale z internetu jak i z książek które podaję na końcu niniejszego opracowania które utworzyłem Sam dla siebie. Stąd może się przytrafić dość duże podobieństwo zdań, wręcz cytatów.

Wskaźniki są kluczowe do dynamicznego zarządzania pamięcią, przekazywania argumentów do funkcji, pracy z tablicami i budowania złożonych struktur danych.

W deklarowaniu wskaźnika, używa się operatorów (&) i (*) . Operator & - służy do pobrania adresu, a * - służy do wyłuskania wartości pod danym adresem. Deklarowanie wskaźnika polega na określeniu typu danych, dodania * oraz nazwy wskaźnika.

Przykład:

```
int *wskaźnik;
```

oznacza:

wskaźnik (*) do zmiennej typu int o nazwie wskaźnik

Aby zainicjować wskaźnik, przypisuje mu się adres innej zmiennej za pomocą operatora - **&.**

Przykład:

a = &c;

oznacza to,

że do uzyskania adresu pamięci zmiennej a , przypisano adres c do wskaźnika a.

Ogólna składnia wskaźnika.

typ_danych *nazwa_wskaznika;

- typ_danych:
  określa, do jakiego rodzaju zmiennej może odnosić się wskaźnik, (np.: int, float, double, char, string, itd.)

- *:
  gwiazdka - oznacza, że deklarowana zmienna jest wskaźnikiem,

- nazwa_wskaznika:
  nazwa, jakiej używa się do odwołania do wskaźnika,

Deklaracja wskaźnika.

Tworząc większy program, jak wiadomo, jeśli większy to i cięższy, zajmujący więcej miejsca, nasuwa się dla mnie myśl, co by zrobić aby ten program nie był za ciężki. I tu dla mnie z pomocą przychodzą właśnie wskaźniki.

Więc deklaruję wskaźniki który nazywam – balast.

int x = 999;

int *balast = &x;

Dlaczego nazwałem wskaźnik – balast? Otóż, inna nazwa by do mojego programu nie pasowała. Właśnie dane które potrzebne mi były do używania w programie, były bardzo ciężkie. Stąd tez pojawiła się nazwa balast. W tym programie który utworzyłem jest również wskaźnik piórko. Dlaczego tak? Nazwa wskaźnika wynikła z cechy wskaźnika, tj. jaką wartość lub do jakiej wartości jest przypisany ten adres na wskaźniku.

Abym w/w wskaźnik balast mógł zainicjalizować, musiał on wskazywać na istniejącą zmienną, w tym przypadku, zmienna jest typu int i który posiada wartość 999. Następnie deklarując wskaźnika balast do zmiennej x której przypisano wartość 999 , przed zmienną x wstawiłem &.

Przykłady deklaracji i inicjalizacji.

1. Wskaźnik do zmiennej całkowitej.

   ```
   int   *wskaźmik_na_int
   ```

2. Inicjalizacja wskaźnika adresem zmiennej.

   aby wskaźnik mógł wskazywać na zmienną, przypisuję mu adres tej zmiennej za pomocą operatora adresu, to jest - &

   ```
   int   wartosc = 999;
   int   *balast_na_int = &wartosc;
   ```

   w tym przykładzie:

   ```
   int   wartosc = 999; // deklaruje zmienną wartosc
   int   *balast_na_int = &wartosc;  /* deklaruje wskaźnik balast_na_int i inicjalizuje go adresem zmiennej wartosc */
   ```

3. Wskaźnik pusty.

   Wskaźnik, który nie wskazuje na żaden obiekt, może być zainicjalizowany wartością nullptr (od C++ 11) lub NULL.

   ```
   int   *wskaźnik_pusty = nullptr;
   ```

Dla przykładu prosty program.

```
/* pierwszy przykładowy wskaźnik – program BALAST

#include <iostream>
<include <conio.h>

using namespace std;

    int main() {

    cout << "program przykladowy BALAST " << endl;
    cout << endl;
    cout << endl;
    cout << "---------- BALAST ----------" << endl;
    cout << endl;

    int   liczba = 999;
    int   *ballast;
    ballast = &liczba;

    cout << "Pokaz zmiennej 'balast': " << liczba << endl;
    cout <<  "Adres zmiennej 'balast': " << &liczba << endl;
```

```
    cout << "Pokaz wartości wskaźnika ( czyli adres 'liczba' ):" << balast << endl;

    cout << "Pokaz wartości na która wskazuje wskaźnik ( 'balast' ):" << *balast << endl;

// a tu cos nowego, czyli zmiana wartości wskaźnika 'balast'
    *balast = 333;

    cout << "Pokaz zmieniona wartosc 'liczba':" << liczba << endl;

    cout << endl;
    cout << endl;

    cout << "------ KONIEC BALASTU ------" << endl;

getch ();
return 0;
}
```

A oto efekt powyższego programu

```
program przykładowy BALAST

--------- BALAST ----------

Pokaz zmiennej 'balast': 999
Adres zmiennej 'balast': 0x6bfef8
Pokaz wartosc wskaźnika ( czyli adres 'liczba'): 0x6bfef8
Pokaz wartosc na ktora wskazuje wskaźnik ( 'balast' ): 999
Pokaz zmieniona wartosc 'liczba': 333

------ KONIEC BALASTU ------
```

W powyższym programie, jak widać została zmieniona wartość wskaźnika z 999 na 333.

Otóż jest to możliwe, i oczywiste jest, że taką czynność można dokonać tak jak powyższy program przedstawił.

Tak dla przypomnienia krótka uwaga.:

Niniejsze opracowanie, jak i następne opracowania celem nauki własnej którą kontynuuję, opieram na lekturach takich jak:

1. Język C++ Szkoła programowania Stephen Prata
2. Programowanie w języku C++ Wiesław Porębski
3. Podstawy języka C++ Stanley B. Lippman
4. Symfonia C++ standard Jerzy Grębosz
5. PRZEWODNIK DLA POCZĄTKUJĄCYCH C++ ALEX ALLAIN
6. Zadania z programowania z przykładowymi rozwiązaniami Mirosław J. Kubiak
7. i internet