Temat: Wskaźniki w języku C++

Wskaźniki i referencje:

- **Wskaźnik** jest to zmienna, która **przechowuje adres innej zmiennej**.
  - * – **operator wyłuskania wartości** zmiennej, na którą wskazuje wskaźnik (wyciąga wartość ze wskaźnika)
  - & – **operator pobrania adresu danej** zmiennej, tablicy, struktury itp. (pobiera adres zmiennej)
- **Referencja** jest to zmienna, która **jest aliasem do prawdziwej zmiennej**. Konstrukcja: typ& nazwa = istniejącaZmienna;
  - Można rozumieć to w taki sposób, że referencja tworzy nową nazwę dla istniejącej zmiennej. Wszystkie zmiany w referencjach są zmianami na prawdziwej zmiennej.

Jak używać wskaźników i referencji:

- Zmienną wskaźnikową (wskaźnik) definiujemy poprzedzając gwiazdką (*) i przechowuje ona adres pamięci (4 bajty) a nie wartość zmiennej,
  - int numer; //zmienna liczbowa (pusta)
    int *wsk; //zmienna wskaźnikowa typu liczbowego (pusty)
- Korzystanie do wskaźnika, który nie wskazuje na żadną ze zmiennych (jak powyżej) prowadzi do błędów, dlatego najlepiej od razu przy definicji przypisać wskaźnikowi adres zmiennej.
  - int numer=123; //zmienna liczbowa
    int *wsk = &numer; //zmienna wskaźnikowa zawiera adres zmiennej numer

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int liczba;           //zmienna liczbowa
7      int *wsk_liczba;      //zmienna wskaźnikowa na int
8
9      liczba = 12;
10     wsk_liczba = &liczba; //przypisanie wskaźnikowi adresu do int
11     liczba = 13;
12     cout<<*wsk_liczba; //wyświetlenie liczby na jaką wskazuje wskaźnik
13     cout<<endl<<liczba<<endl;
14     liczba = 999;
15     cout<<*wsk_liczba<<endl;
16     cout<<liczba<<endl;
17     return 0;
18 }
```

```
13
13
999
999
```

Wskaźniki, referencje i funkcje:

- W języku C i C++ możemy przekazywać wartości do funkcji przez przekazywanie przez wskaźnik. Wskaźnik będzie wtedy argumentem funkcji.

```
1   #include <iostream>
2
3   using namespace std;
4
5   void pomnozRazyDwa(int *liczba){
6       *liczba = *liczba *2;
7   }
8
9   int main(){
10      int liczba = 5;
11      int *adres = &liczba;
12
13      cout<<"Przed pomnozeniem "<<liczba<<endl;
14      pomnozRazyDwa(adres);
15      cout<<"Po pomnozeniu "<<liczba<<endl;
16
17
18      return 0;
19  }
20
```

Można zastąpić przez `pomnozRazyDwa(&liczba);`

```
Przed pomnozeniem 5
Po pomnozeniu 10
```

- Jako argument funkcji można pobierać również referencję do oryginalnej zmiennej, wtedy zmiany będą dokonywane na istniejących zmiennych

```
1   #include <iostream>
2
3   using namespace std;
4
5   int pomnoz(int &a, int &b){
6       return a*b;
7   }
8
9   int main(){
10      int u = 5;
11      int w = 7;
12      cout<<"Po pomnozeniu "<<pomnoz(u,w)<<endl;
13
14      return 0;
15  }
16
```

```
Po pomnozeniu 35
```

Wskaźniki i tablice:

- Co tak naprawdę oznacza używanie operatora [...] poznanego przy okazji tablic?

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main()
6   {
7       int Tab[10];
8       cout<<"Tab wskazuje na adres pierwszego elementu: "<<Tab<<endl;
9
10      *(Tab+4) = 5; //Równoważne zapisowi Tab[4]=5;
11
12      cout<<"Wartosc Tab[4]="<<Tab[4];
13  }
14
```

```
Tab wskazuje na adres pierwszego elementu: 0x61fdf0
Wartosc Tab[4]=5
Process returned 0 (0x0)   execution time : 0.944 s
Press any key to continue.
```

- Poniżej znajdują się trzy równoważne wersje odczytania wartości z tablicy.

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main()
6   {
7       int Tab[10]= {1,2,3,4,5,6,7,8,9};
8       int *Wsk=Tab;
9
10      for(int i=0; i<10; i++)
11          cout<<Wsk[i]<<" ";
12
13      cout<<endl;
14
15      for(int i=0; i<10; i++)
16      {
17          cout<<*(Tab+i)<<" ";
18      }
19
20      cout<<endl;
21
22      for(int i=0; i<10; i++)
23      {
24          cout<<*Wsk<<" ";
25          Wsk++;
26      }
27  }
28
```

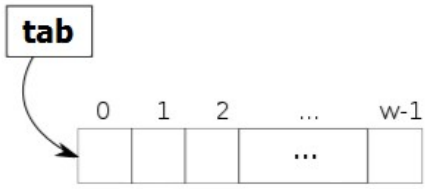
W tym miejscu program za każdym razem musi wykonywać operację dodawania ***(Wsk+i)**, która jest wolniejsza od czystej inkrementacji przedstawionej w ostatniej pętli for.

```
1 2 3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8 9 0
```

Dynamiczna alokacja pamięci:

- Dynamiczne alokowanie pamięci pozwala na tworzenie i zwalnianie pamięci zawartej w zmiennej w dowolnym momencie.
- Do alokowania pamięci służy operator **new**, a zwalniania **delete**.
- Utworzona zmienna (nawet wewnątrz definicji funkcji) pozostaje w pamięci do czasu aż ją usuniemy.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int w=10;
7      double* tab = new double[w]; //alokacja pamieci
8
9      for(int i=0;i<w;++i)
10         tab[i]=2*i;
11
12     for(int i=0;i<w;++i)
13         cout<<tab[i]<<", ";
14
15     delete [] tab;                //uwolnienie pamieci
16     tab = NULL;
17
18     return 0;
19 }
```




```
0, 2, 4, 6, 8, 10, 12, 14, 16, 18,
```

Zadania do samodzielnej realizacji:

1. (2p) Pobierz od użytkownika wartości dwóch zmiennych typu int. Napisz funkcję **void zamianaWartosci**, której parametrami będą dwa wskaźniki do zmiennych typu int. Zadaniem funkcji jest zamiana wartości zmiennych (nie wystarczy tylko wyświetlenie ich w odwrotnej kolejności). Wyświetl wartości przed zamianą i po niej.

2. (2p) Napisz funkcję **int* tablica(int n)**, której argumentem jest dodatnia liczba całkowita **n**. Funkcja tworzy dynamiczną jednowymiarową tablicę elementów typu int o **n** losowo wybranych wartościach z przedziału [-10,10]. Funkcja zwraca wskaźnik do pierwszego elementu tablicy. Wyświetl wartości tablicy. **n** podaje użytkownik.

3. (1p) Stwórz plik o nazwie macierz.txt za pomocą klasy fstream. Napisz funkcję **void losuj**, której argumentami są: referencja do obiektu klasy fstream oraz zmienna całkowitoliczbowa int n. Zadaniem funkcji jest uzupełnienie pliku macierzą pseudolosową wymiaru n x n liczbami całkowitymi z przedziału [0, 9]. Wartość n ma podać użytkownik.