Wskaźniki w C

Wyobraźmy sobie pamięć komputera, jako taką schematyczną kratownicę. W polach kratownicy zapisywane są zmienne, które zostały zdefiniowane w programie. Każda kratka ma swój unikalny adres, dzięki któremu możemy odczytać wartość zmiennej. Wskaźniki przechowują właśnie te adresy.

Wskaźnik to zmienna zawierająca adres jakiejś zmiennej (lub początkowej komórki tablicy) w pamięci programu. Adres zmiennej to jej nazwa poprzedzona znakiem &. Adres tablicy (a właściwie jej komórki nr 0), to nazwa tej tablicy. Adres każdej kolejnej komórki jest większy o 1 od adresu poprzedniej. Typ wskaźnika to typ zmiennej, której adres ma przechowywać, wraz ze znakiem *.

Przykład utworzenia zmiennej typu int i wskaźnika na nią:

```
int a; // definicja wskaźnika
int* wsk;  // definicja wskaźnika
wsk = &a;  // ustawienie wskźnika na komórke ze zmienna a
```

Przykład utworzenia tablicy typu char i wskaźnika na nią:

```
char tab[10];
char(*ptr)[10];  // ptr to wskaźnik na tablicę 10 znaków
ptr = tab;  // teraz ptr wskazuje na tab
```

Dostęp do zmiennej przez wskaźnik

Dostęp do zawartości zmiennej za pośrednictwem wskazującego na nią wskaźnika uzyskuje się przez nazwę wskaźnika poprzedzoną znakiem *, np.:

```
int* ptr; // definicja wskaznika
int a = 5;
ptr = &a; // ustawiamy wskaźnik na komórkę ze zmienną numer
printf("\nWartosc zmiennej a = %d", *ptr); // przed nazwą wskaźnika dajemy gwiazdkę
```

Przykład 1.

```
int i, * p = &i;
for (i = 0; i < 10; i++)
      printf("%d\n", *p);
```

Podany przykład powoduje wyświetlenie liczb od 0 do 9, ale nie posługuje się w żadnym wypadku zmienną i, lecz wskaźnikiem do niej. Gwiazka * postawiona przy wskaźniku p informuje że w tym miejscu kompilator ma podstawić wartość zmiennej na której adres wskazuje. Wskaźnik wywołany bez gwiazdki wskazuje na adres zmiennej w pamięci, nie na wartość!

Zastanów się jak wyglądałoby wywołanie funkcji scanf(). Oczywiście postawilibyśmy tak wskaźnik bez gwiazdki, ponieważ używając tej funkcji zapisujemy wartość pod adresem w pamięci, co można było zauważyć już wcześniej, gdzie zmienna w funkcji scanf() miała postać &zmienna.

Przykład 2.

```
    int i, * wsk, a[5] = { 1,2,3,4,5 };
    wsk = a;
    for (i = 0; i < 5; i++)
        printf("%d\n", a[i]);
    printf("\n");
    for (i = 0; i < 5; i++)
        printf("%d\n", *(wsk + i));
    printf("\n");
    for (i = 0; i < 5; i++)
        printf("%d\n", wsk[i]);
```

Przykład 3.

```
int main()
{
    char* wsk, ph[100];
    printf("wprowadz fraze\n");
    gets(ph);
    wsk = ph;
    while (*wsk)
    {
        *wsk = toupper(*wsk);
        *wsk++;
    }
    printf("%s\n", ph);
}
```

Przykład 4.

```
int main()
{
    char* a = "koniec";
    char tablica[30];
    do
    {
        printf("wpisz slowo\n");
        gets(tablica);
    } while (strcmp(a, tablica));
}
```

Przykład 5.

```
int main()
{
      int i, j, punkty[4][5];
      for (i = 0; i < 4; i++)
      {
            for (j = 0; j < 5; j++)
            {
                  printf("%d kwarta, %d zawodnik\n", i + 1, j + 1);
                  printf("ile punktow?\n");
                  scanf("%d", *(punkty + i) + j);
            }
      }
      printf("\n\n");
      for (i = 0; i < 4; i++)
      {
            for (j = 0; j < 5; j++)
            {
                  printf("%d kwarta, %d zawodnik", i + 1, j + 1);
                  printf(" %d punktow\n", *(*(punkty + i) + j));
            }
      }
```

```
}
```

Przykład 6.

```
int main()
{
    int i;
    char polski[20], * slowa[10][2] =
    { "mama","mother",
    "tata","father",
    "dom","house",
    "pilka","ball",
    "plaza","beach",
    "niebo","sky",
    "szkola","school",
    "muzyka","music",
    "chlopiec","boy",
    "rower","bike" };
    printf("wpisz polskie slowo\n");
    gets(polski);
    for (i = 0; i < 10; i++)
    {
        if (!strcmp(polski, slowa[i][0]))
        {
            printf("%s- %s\n", polski, *(*(slowa + i) + 1));
            break;
        }
    }
}
```

Przykład 7.

```
int main()
{
    int a, * wsk, ** wsk2;
    wsk = &a;
    wsk2 = &wsk;
    printf("podaj wartosc\n");
    scanf("%d", *wsk2);
    printf("wartosc zmniennej a: %d\n", **wsk2);
}
```

Wskaźniki i funkcje:

```
void ZwiekszLiczbe(int* liczba)
{
      *liczba += 5;
}

int main()
{
      int numer = 5;
      int* wsk = &numer;
      ZwiekszLiczbe(wsk); //przekazujemy wskaźnik (bez operatorów)
      printf("\n%d", numer);
      ZwiekszLiczbe(&numer); //przekazujemy bezpośrednio adres zmiennej (operator
&)
      printf("\n%d", numer);
}
```

Przekazywanie argumentów przez wskaźnik

Przekazywanie argumentów przez wskaźnik jest możliwe zarówno w języku C++ jak i C. Użycie wskaźników razem z funkcjami jest bardzo wygodne i ekonomiczne. Zdarzają się sytuacje, kiedy skorzystanie ze wskaźników jest niezbędne do osiągnięcia odpowiedniego efektu.

Przekazując argumenty przez wartość, funkcja może zwrócić tylko jeden parametr za pomocą return. Oznacza to, że możliwe jest zmodyfikowanie wartości tylko jednej zmiennej występującej poza wywoływaną funkcją. Co w przypadku gdy podczas wywołania jednej funkcji chcemy zmienić kilka zmiennych? – tutaj niezbędne są wskaźniki. Jest to najlepszy przykład na zobrazowanie tego, dlaczego wskaźniki są przydatne.

Podsumowując przekazywanie przez wskaźnik:

- argumenty wskaźnikowe wskazują na zmienne z poza funkcji, więc wewnątrz funkcji nie są tworzone kopie
- funkcja może zmodyfikować wiele zmiennych z poza funkcji (bez użycia return)
- wskaźniki także przekazujemy przez wartość, wynika to z faktu że wskaźnik też jest typem zmiennej (zmienna wskaźnikowa), jednak mimo że wskaźnik (adres miejsca w pamięci) zostanie skopiowany to wartość wyłuskana ze wskaźnika nie jest już kopią

```
void zwieksz_kilka(int* dl, int* wys, int* waga)
{
	// zmienna '*dl', '*wys' i '*waga' nie są kopiami
	// operowanie na nich zmienia ich wartość w "całym" programie
	// funkcja nie zwraca nic - bo nie ma sensu

	*dl = *dl * 2;
	*wys = *wys * 2;
	*waga = *waga * 2;
}

int main()
{
	// zmienne
	int dlugosc = 125;
	int wysokosc = 300;
	int waga = 20;

	// wskaźniki do zmiennych
	int* wsk_dlugosc = &dlugosc;
	int* wsk_wysokosc = &wysokosc;
	int* wsk_waga = &waga;

	// wywołanie funkcji
	zwieksz_kilka(wsk_dlugosc, wsk_wysokosc, wsk_waga);

	// wyświetlenie nowych wartości
	printf("\n%d", dlugosc);
	printf("\n%d", wysokosc);
	printf("\n%d", waga);
```

```
}
```

Zwróć uwagę, argumenty funkcji to zmienne wskaźnikowe czyli „adresy do zmiennych”. Gwiazdka w argumentach nie jest operatorem wyłuskania, informuje ona kompilator że zmienna jest wskaźnikiem (czyli „adresem”). Jak można wykorzystać ten fakt? Nie trzeba deklarować zmiennych wskaźnikowych aby przekazywać argument funkcji przez wskaźnik. Można przekazać bezpośrednio adres za pomocą operatora pobrania adresu (&).

```
void zwieksz_kilka(int* dl, int* wys, int* waga)
{
	// zmienna '*dl', '*wys' i '*waga' nie są kopiami
	// operowanie na nich zmienia ich wartość w "całym" programie
	// funkcja nie zwraca nic - bo nie ma sensu

	*dl = *dl * 2;
	*wys = *wys * 2;
	*waga = *waga * 2;
}

int main()
{
	// zmienne
	int dlugosc = 125;
	int wysokosc = 300;
	int waga = 20;

	// wywołanie funkcji z (&) (tak jak byśmy przekazywali wskaźniki)
	zwieksz_kilka(&dlugosc, &wysokosc, &waga);

	// wyświetlenie nowych wartości
	printf("\n%d", dlugosc);
	printf("\n%d", wysokosc);
	printf("\n%d", waga);

}
```

Zadania do samodzielnego wykonania:

Wykorzystując wskaźniki należy zaproponować implementację do poniższych zadań.

1. Napisz program, który:
   - wypisze na ekran adres zadeklarowanej zmiennej,
   - pobierze wartość zmiennej, wypisze na ekran jej adres oraz wartość,
   - przy użyciu wskaźników obliczy różnicę dwóch liczb,
   - przy użyciu wskaźników obliczy średnią trzech liczb,
2. Napisz program, w którym zadeklarujesz tablicę a następnie:
   - Wypisz na ekran adres jej pierwszego elementu.
   - Wypisz na ekran adres jej czwartego elementu.
   - Napisz program, w którym wylosujesz wartości do tablicy z przedziału podanego przez użytkownika i wypiszesz ich adresy.
3. Napisz program, w którym wczytasz elementy tablicy, obliczysz ich średnią oraz wpiszesz elementy większe od średniej.
4. Napisz funkcję, która podnosi podaną liczbę do kwadratu. Parametr do funkcji przekaż poprzez wskaźnik.

5. Napisz funkcję, która umożliwi policzenie podanej potęgi dla podanej liczby. Parametry przekaż poprzez wskaźnik. Funkcja powinna zwrócić wskaźnik do wyniku.
6. Napisz funkcję, która wypisuje podany znak podaną liczbę razy. Parametry przekaż poprzez wskaźniki. Na zakończenie funkcja powinna zmniejszać o 1, parametr odpowiadający za ilość powtórzeń.
7. Napisz funkcję otrzymującą jako argumenty wskaźniki do dwóch zmiennych typu int, która:
   - zwraca jako wartość mniejszą z liczb wskazywanych przez argumenty.
   - zwraca jako wartość wskaźnik na zmienną przechowującą mniejszą z liczb wskazywanych przez argumenty.
8. Napisz funkcję otrzymującą jako argumenty referencje do dwóch zmiennych typu int, która zamienia ze sobą wartości zmiennych, do których referencje dostaliśmy w argumentach.