C - 6 - Wskaźniki i napisy

6.1 Definicja i proste przykłady

Wskaźnik, to rodzaj zmiennej, która przechowuje adres innej zmiennej – wskazuje więc jej adres. W dotychczasowych przykładach przekazywaliśmy argumenty do funkcji przez **wartość** – funkcja dostawała więc „kopię" wartości zmiennej. Od tej chwili będziemy też przekazywać argumenty przez **referencję** – funkcja dostanie adres zmiennej (chodzi o adres w pamięci komputera), pójdzie pod wskazany adres, zobaczy wartość zmiennej i ewentualnie coś z tą wartością zrobi.

Wskaźniki odróżniamy od zwykłych zmiennych dzięki symbolom * oraz &. Przykład:

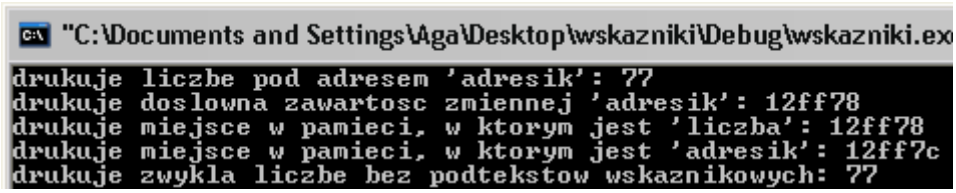
```
#include "stdafx.h"

int main(int argc, char* argv[]){

        int  *adresik;
        int liczba = 77;

        adresik = &liczba;

        printf("drukuje liczbę pod adresem 'adresik': %d \n",*adresik);
        printf("drukuje dosłowną zawartość zmiennej 'adresik': %x \n",adresik);
        printf("drukuje miejsce w pamięci, w ktorym jest 'liczba': %x \n",&liczba);
        printf("drukuje miejsce w pamięci, w ktorym jest 'adresik': %x \n",&adresik);
        printf("drukuje zwykłą liczbę bez podtekstów wskaźnikowych: %d \n",liczba);

        return 0;
}
```

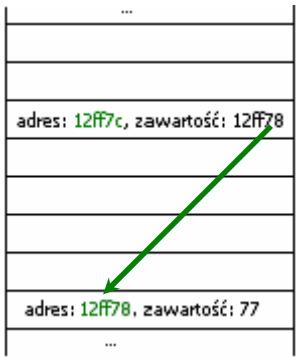
Można też napisać int* adresik; - gwiazdka może być przy typie lub przy nazwie zmiennej.
Instrukcja ta oznacza deklarację „kartki na adres" zmiennej typu całkowitoliczbowego.

Zwykła zmienna bez podtekstów wskaźnikowych :)

Ta instrukcja oznacza: „od tej chwili niech zmienna wskaźnikowa adresik przechowuje adres zmiennej liczba"

Powyższy kod da różne rezultaty na różnych komputerach – zależnie od zawartości pamięci. Na moim komputerze kod zadziałał następująco:





Zadanie 0. Zadeklaruj dwie zmienne wskaźnikowe różnych typów i przypisz im adresy różnych zmiennych. Wyświetl informacje podobne do powyższych, następnie narysuj na kartce schemat pamięci.

Przykład funkcji pobierającej wskaźnik do zmiennej:

```
#include "stdafx.h"

void pomnozRazyDwa(int *adr){
        *adr = *adr * 2;
}

int main(int argc, char* argv[]){

        int *adresik;
        int liczba = 77;

        adresik = &liczba;

        printf("Przed: %d ...",*adresik);
        pomnozRazyDwa(adresik);
        printf("... i po: %d \n\n",*adresik);

        return 0;
}
```

Ta funkcja nic nie zwraca, tylko idzie pod wskazany adres, mnoży razy dwa i wraca na swoje miejsce :)
Niebieskie gwiazdki są wskaźnikowe, zielona gwiazdka to zwvkłe mnożenie.

Podajemy adres zmiennej liczba (bez * i & !!!), funkcja idzie pod ten adres i mnoży.

Kolejny przykład – tym razem napiszemy funkcję, która zamienia miejscami zawartość dwóch zmiennych. Mówiąc dokładniej, zamienimy adresy:

```
#include "stdafx.h"

void zamien(int *px, int *py) {
		int pom;
		pom = *px;
		*px = *py;
		*py = pom;
}

int main(int argc, char* argv[]){
		int x = 77;
		int y = 109;

		printf("Przed: %d %d...",x,y);
		zamien(&x,&y);
		printf("... i po: %d %d\n\n",x,y);

		return 0;
}
```

Do funkcji przekazujemy tylko „kartki z adresami”. Funkcja wymaże na nich dotychczasowe adresy i wpisze je na przeciwnych kartkach.

Często będziemy potrzebowali wskaźnik do pierwszego elementu tablicy:

```
#include "stdafx.h"

int main(int argc, char* argv[]){

		int tab[10]={9,2,3,4,7,8,2,6,2,0};
		int *poczatek;

		poczatek = &tab[0];

		printf("Pierwszy element tablicy: %d \n",*poczatek);
		poczatek ++;
		printf("Drugi element tablicy: %d \n",*poczatek);
		poczatek ++;
		printf("Trzeci element tablicy: %d \n",*poczatek);
		// i tak dalej ...

		return 0;
}
```

Patrzymy na „kartkę z adresem” i zwiększamy „numer mieszkania” o jeden.

Zadanie 1. Przerób którąś z wersji sortowania bąbelkowego tak, by do zamiany dwóch elementów służyła osobna funkcja działająca na wskaźnikach.

Wskaźniki do któregoś elementu tablicy przydadzą się najbardziej podczas operacji na napisach. Przypominam, że napis kończy się niewidzialnym znaczkiem **\0** !!! Większy przykład ilustrujący wiele możliwości operacji na napisach:

```
#include "stdafx.h"

int dlugoscNapisu(char *s){

		int licznikLiterek = 0;
		while(*s !='\0'){
			s ++;
			licznikLiterek ++;
		}
		return licznikLiterek;
}

int main(int argc, char* argv[]){

		char napis[10]="bleh...";
		char innyNapis[10] = "alamakota";
		char *wskaznik = &innyNapis[0];
		char* wyraz = "alamakotka";

		int test1 = dlugoscNapisu(napis);
		int test2 = dlugoscNapisu("Ahoj przygodo !!!");
		int test3 = dlugoscNapisu(wskaznik);
		int test4 = dlugoscNapisu(wyraz);

		printf("test1: %d\n",test1);
		printf("test2: %d\n",test2);
		printf("test3: %d\n",test3);
		printf("test4: %d\n",test4);
		return 0;
}
```

Dopóki nie natrafimy na zakończenie wyrazu, to:
- idziemy o komórkę dalej
- zwiększamy licznik literek

Różne możliwości argumentów dla tej samej funkcji:
- cała tablica,
- „dosłowny” napis w cudzysłowach,
- wskaźnik do pierwszego elementu,
- wskaźnik z wartością przypisaną podczas deklaracji.

Zadanie 2. Napisz podobną funkcję zliczającą samogłoski w wyrazie (małe i duże).

Zadanie 3. Napisz podobną funkcję zamieniającą wszystkie małe litery na duże.

6.2 Operacje na napisach

Przypomnienie: do wyświetlania napisów (czyli stringów) używaliśmy **%s**.

```
#include "stdafx.h"

int main(int argc, char* argv[]){
	char* wyraz = "Ala ma kota o imieniu \"Cezar\" ";
	printf("Twoja wiadomosc, to: %s.\n",wyraz);
	return 0;
}
```

Przy okazji mamy przykład pisania cudzysłowów wewnątrz innych cudzysłowów.

Przykład pobrania tekstu z klawiatury:

```
#include "stdafx.h"

int main(int argc, char* argv[]){
	char wyraz[256];
	printf("Wpisz jakiś tekst i naciśnij ENTER:\n");
	gets(wyraz);
	printf("Twoja wiadomosc, to: %s.\n",wyraz);
	return 0;
}
```

Przykład pobrania określonej liczby wyrazów:

```
#include "stdafx.h"

int main(int argc, char* argv[]){
	char imie[256], nazwisko[256];
	printf("Wpisz swoje imie i nazwisko i nacisnij ENTER:\n");
	scanf("%s %s",imie,nazwisko);
	printf("Nazywasz sie: %s %s.\n",imie,nazwisko);
	return 0;
}
```

Tu wyjątkowo nie ma & przed nazwą zmiennej w scanfie !!!

W języku C/C++ istnieje wiele wbudowanych funkcji działających na napisach. Znajdują się one w bibliotece **string.h** – nie zapomnij jej zaincludować !!!

Funkcja **int strlen(char*)** zwraca długość wyrazu – działa identycznie jak napisana powyżej funkcja **dlugoscWyrazu**.

Zadanie 4. Napisz program, który wczytuje z klawiatury linijki tekstu oddzielone ENTERem – po każdej linijce program ma wypisywać długość wpisanej linijki. Jeżeli wprowadzono pustą linijkę (czyli ENTER zamiast tekstu), program kończy swoje działanie. Przyda się pętla **while** oraz funkcje **gets** i **strlen**.

Funkcja **char *strcpy(char *dokad, char *skad)** kopiuje napis z jednej zmiennej do drugiej. Przykład:

```
#include "stdafx.h"
#include "string.h"
#include "stdlib.h"

int main(int argc, char* argv[]){

	char skad[] = "To jest napis zrodlowy.";
	char dokad[80];
	char *test1;

	printf("Chcemy przekopiować: %s\n",skad);

	//mozemy bez zadnych problemow przekopiowac tablice do tablicy:
	strcpy(dokad,skad);
	printf("Teraz w dokad mamy : %s\n",dokad);

	//ale żeby przekopiować do wskaznika, musimy mu przydzielic pamiec:
	test1 = (char *)malloc(strlen(skad) +1);
	strcpy(test1, skad);
	printf("Teraz w test1 mamy: %s\n", test1);

	return 0;
}
```

Ta biblioteka zawiera funkcję malloc, która przydziela zmiennej odpowiednią ilość pamięci. malloc to skrót od „memory allocate"

Zauważ, że tu nie podaliśmy rozmiaru tablicy w nawiasach kwadratowych a program i tak działa :)

Sprawdź, jak zadziała program bez tej linijki.

Jeżeli chcemy przekopiować tylko **n** pierwszych znaków, to skorzystamy z funkcji
char *strncpy(char *destination, char *source, size_t n)

Zadanie 5. Przerób powyższy program – niech użytkownik poda z klawiatury ile znaków chce przekopiować. Jeżeli liczba znaków (**strlen** !!!) będzie większa od długości napisu w tablicy **skad**, to przekopiuj całą tablicę.

Do skonkatenowania (czyli sklejenia napisów) służą funkcje:
char *strcat(char *str1, char *str2) oraz **char *strncat(char *str1, char *str2, size_t n)**
Przykład:

```
#include "stdafx.h"
#include "string.h"

int main(int argc, char* argv[]){

	char str1[27];
	char str2[] = "abcdefghijklmnopqrstuvwxyz";
	char str3[] = "123456789";
	char str4[] = "!@#$^&*";

	// przypisujemy pusty napis do str1
	// żeby wykasować ewentualne śmieci w pamięci
	strcpy(str1,"");

	// doklejamy różne napisy do str1:
	strcat(str1,str3);
	strcat(str1,str4);
	printf("str1 po przerobkach: %s\n",str1);

	// znowu czyścimy str1:
	strcpy(str1,"");

	// doklejamy tylko 9 znaków z napisu str2:
	strncat(str1,str2,9);
	printf("str1 po kolejnych przerobkach: %s\n",str1);

	return 0;
}
```

Zadanie 6. Przerób powyższy program – w jednej pętli **for** doklejaj coraz większą liczbę znaków z str2 do str1 i wyświetlaj zawartość na ekranie tak, by powstała piramida:
a
ab
...
abcdefghijklmnopqrstuvwxyz

Do leksykograficznego porównywania dwóch napisów użyjemy funkcji
int strcmp(char *str1, char *str2) oraz **int strncmp(char *str1, char *str2, size_t n)**
Funkcje ta zwracają wartości: 0, jeżeli napisy są identyczne
>0, jeżeli str1 > str2
<0, jeżeli str1 < str2

Zadanie 7. Napisz program, który będzie pobierał (w nieskończonej pętli) z klawiatury po dwa wyrazy podane przez użytkownika. Program ma wyświetlać te dwa wyrazy w kolejności leksykograficznej. Jeżeli jeden z wyrazów będzie brzmiał **exit**, to program powinien zakończyć działanie.

Zadanie 8. Przerób program z zadania 7 – niech użytkownik podaje, ile pierwszych liter należy porównać w wyrazach.

Poniższa funkcja znajduje wskaźnik do pierwszego wystąpienia danego znaku w wyrazie:
char *strchr(char *str, int ch)
Przykład:

```
#include "stdafx.h"
#include "string.h"

int main(int argc, char* argv[]){

	char wyraz[] = "abrakadabra";
	char *szukany;
	int znaczek = 'k';
	szukany = strchr(wyraz,znaczek);
	if(szukany==NULL)  printf("Nie ma takiej literki.\n");
	else printf("Znaczek %c znajduje sie na pozycji %d liczac od zera.\n", znaczek, szukany-wyraz);
	return 0;
}
```

Odejmujemy wskaźniki !!!

Zadanie 9. Korzystając z poznanych funkcji stringowych napisz program, który znajdzie każde wystąpienie danej litery w wyrazie. W tym zadaniu nie wolno po prostu przeglądać wyrazu literka po literce, trzeba koniecznie skorzystać z przynajmniej jednej poznanej funkcji o nazwie zaczynającej się od **str**.

Funkcja **char *strrchr(char *str, int ch)** znajduje ostatnie wystąpienie danego znaku w wyrazie.

Zadanie 10. Przerób zadanie 9 korzystając z funkcji **strrchr**.

Funkcja **char *strstr(char *str1, char *str2)** znajduje wskaźnik do pierwszego wystąpienia wyrazu **str2** w **str1**. Przykład:

```
#include "stdafx.h"
#include "string.h"

int main(int argc, char* argv[]){

	char wyraz1[] = "abrakadabra";
	char wyraz2[] = "ra";
	char *szukany;

	szukany = strstr(wyraz1,wyraz2);
	if(szukany==NULL) printf("Wyraz2 nie zawiera sie w wyrazie1.\n");
	else printf("Wyraz %s znajduje się w wyrazie %s na pozycji %d.\n", wyraz2,wyraz1, szukany-wyraz1);
	return 0;
}
```

Zadanie 11. Napisz program, w którym użytkownik poda wyraz i podwyraz, program powinien znaleźć każde wystąpienie podwyrazu w wyrazie.

Zadanie 12. Napisz własną funkcję **znajdowaczPodwyrazow** działającą identycznie jak **strstr**.

Funkcje **char *strlwr(char *str)** oraz **char *strupr(char *str)** zamieniają wielkość liter na małe lub duże.

Zadanie 13. Napisz program ilustrujący działanie dwóch powyższych funkcji.

Zadanie 14. Przeczytaj w internecie o funkcjach:
a) **size_t strcspn(char *str1, char *str2)**
b) **size_t strspn(char *str1, char *str2)**
c) **char *strpbrk(char *str1, char *str2)**
d) **char *strrev(char *str)**
e) **char *strset(char *str, int ch)**
f) **char *strnset(char *str, int ch, size_t n)**
i napisz program ilustrujący ich działanie. Nie pomiń żadnej funkcji.

6.3 Zamiana napisy <-> liczby

W bibliotece **stdlib.h** znajduje się funkcja zamieniająca napis na liczbę: **int atoi(char *ptr)**. Jeżeli w napisie znajduje się jakiś nieoczekiwany znak, np. litera, to funkcja zwraca wartość zero. Jej nazwa to skrót od „ascii to integer”. Przykład:

```
#include "stdafx.h"
#include "stdlib.h"

int main(int argc, char* argv[]){
	char wyraz1[] = "12312";
	char wyraz2[] = "-4535";
	char wyraz3[] = "+4789";
	printf("%d %d %d\n",atoi(wyraz1),atoi(wyraz3),atoi(wyraz3));
	printf("Razem: %d\n",atoi(wyraz1)+atoi(wyraz3)+atoi(wyraz3));
	return 0;
}
```

Zadanie 15. Napisz program ilustrujący działanie funkcji **atof**, która zamienia napisy na liczby typu **double**.

Funkcja **char* itoa(int liczbaDoZamiany, char * napis, int podstawa)** działa odwrotnie do funkcji **atoi** – zamienia liczbę na napis. Jako trzeci parametr podaje się podstawę systemu liczbowego (10 – dziesiętne, 2 – binarnie, itd ...). Przykład:

```
#include "stdafx.h"
#include "string.h"
#include "stdlib.h"

int main(int argc, char* argv[]){

        char buf[256];
        strcpy(buf,"");

        itoa(17,buf,10);
        printf("Dziesietnie: %s\n",buf);

        itoa(17,buf,2);
        printf("Binarnie: %s\n",buf);

        return 0;
}
```

Zadanie 16. Napisz program korzystający z funkcji **itoa**, wyświetlający jakąś dużą liczbę (około 50000) w systemach od binarnego do dwudziestkowego. Nie wypisuj kilkunastu **printf**ów, skorzystaj z pętli.

6.4 Funkcje działające na znakach

Poniższa tabela przedstawia funkcje boolowskie testujące właściwości podanego znaku. Funkcje te znajdują się w bibliotece **ctype.h**.

bool isalnum(int znaczek)	Zwraca true, jeżeli znaczek jest literą lub cyfrą.
bool isalpha(int znaczek)	Zwraca true, jeżeli znaczek jest literą.
bool isascii(int znaczek)	Zwraca true, jeżeli znaczek ma kod ASCII od 0 do 127.
bool isdigit(int znaczek)	Zwraca true, jeżeli znaczek jest cyfrą.
bool isgraph(int znaczek)	Zwraca true, jeżeli znaczek jest widzialny na ekranie (np. klawisz F1 nie jest widzialny). Funkcja ta nie uwzględnia spacji.
bool islower(int znaczek)	Zwraca true, jeżeli znaczek jest małą literą.
bool isupper(int znaczek)	Zwraca true, jeżeli znaczek jest dużą literą.
bool isspace(int znaczek)	Zwraca true, jeżeli znaczek jest białym znakiem (spacja, tabulacja, enter, itd...)
bool ispunct(int znaczek)	Zwraca true, jeżeli znaczek jest znakiem interpunkcyjnym.
bool isprint(int znaczek)	Zwraca true, jeżeli znaczek jest znakiem drukowalnym. Uwzględnia spacje.
bool isxdigit(int znaczek)	Zwraca true, jeżeli znaczek jest cyfrą szesnastkową.

Zadanie 17. Napisz program, który w pętli przejrzy wszystkie znaki ASCII od 0 do 255 i przy każdy wypisze jego opis, korzystając ze wszystkich funkcji **is...(...)** zawartych w powyższej tabeli. Przykłady:

A – litera lub cyfra, litera, kod ASCII od 0 do 127, widzialny, duża litera, drukowalny, cyfra szesnastkowa.

, - kod ASCII od o do 127, widzialny, znak interpunkcyjny, drukowalny

6.5 Tablice napisów

Skoro napis jest tablicą jednowymiarową, to tablica napisów będzie dwuwymiarowa. Przykład:

```
#include "stdafx.h"

int main(int argc, char* argv[]){
        char *miasta[3] = {"Gdansk","Warszawa","Ulan-Bator"};

        printf("Eksperymenty:\n\n");
        for(int i=0;i<3;i++) printf("%d) %s\n",i+1,miasta[i]);

        printf("\n");

        printf("\n'*miasta'       -> %s",*miasta);
        printf("\n'*(miasta+1)'   -> %s",*(miasta+1));
        printf("\n'*miasta+1'     -> %s",*miasta+1);
        printf("\n'*(miasta+2)+1' -> %s",*(miasta+2)+1);
        printf("\n'*(miasta[1])'  -> %c",*(miasta[1]));
        printf("\n'*(miasta[1]+3)' -> %c",*(miasta[1]+3));
        printf("\n'miasta[1][3]'  -> %c",miasta[1][3]);
        printf("\n'**miasta'      -> %c",**miasta);

        printf("\n");
        return 0;
}
```

Zadanie 18. Przerób program tak, by wypisywał:

a) napisy **Bator**, **sk**, **szawa**
b) trzy ostatnie litery każdego wyrazu – w pętli **for** należy wydrukować zawartość wskaźnika od pozycji **strlen(...)**-3
c) nazwę pierwszego miasta – każdą literkę w osobnej linijce

Przykład wczytywania wyrazów z klawiatury do tablicy:

```
#include "stdafx.h"

int main(int argc, char* argv[]){

        char miasta[3][20];

        for(int i=0;i<3;i++){
                printf("podaj wyraz (%d) w celu zapisania go do tablicy: ",i);
                gets(miasta[i]);
        }

        printf("Podales wyrazy:\n\n");

        for(i=0;i<3;i++) printf("%d) %s\n",i+1,miasta[i]);

        printf("\n");

        return 0;
}
```

Zadanie 19. Napisz program, w którym użytkownik poda ile (<=10) wyrazów chce posortować, a następnie poda te wyrazy. Program powinien posortować wyrazy bąbelkowo – należy skorzystać z funkcji **strcmp**. Można przyjąć, że wyrazy mają co najwyżej 20 znaków.

6.6 Najczęstsze błędy

Nie includowanie potrzebnych bibliotek.
Mylenie **&** oraz ***** lub ich nieużywanie.
Przerabianie wyrazów literka po literce zamiast stosowania funkcji ze **string.h**.
Zapominanie o **" "** i **' '** dookoła wyrazów i pojedyńczych znaczków.

6.7 Quiz

1. Co się pokaże na ekranie i dlaczego?

```
int tab[10]={19,21,23,24,37,48,52,56,25,70};
int *poczatek;
poczatek = &tab[0];
poczatek = poczatek+7;
printf("%d",*poczatek);
```

_____
_____
_____

2. Wymień po dwie funkcje znajdujące się w podanych bibliotekach i opisz własnymi słowami, co one robią:

string.h

_____
_____
_____
_____
_____
_____

stdlib.h

_____
_____
_____
_____
_____
_____

ctype.h

_____
_____
_____
_____
_____
_____

3. Co robią poniższe funkcje? Opisz ich działanie własnymi słowami.

strcspn

_____
_____

strspn

_____
_____

strpbrk

_____
_____

strrev

_____
_____

strset

_____
_____

strnset

_____
_____

4. W jaki sposób można zapisać w środku napisu cudzysłowy? _____

5. W jaki sposób należy kopiować napis do wskaźnika?

_____
_____

6. Dana jest tablica **char* jedzenie[4] = {"dzik z rozna","pumpernikiel","kefirek","guma do zucia"};**

Co pojawi się na ekranie i dlaczego?

printf("\n%s",*jedzenie); _____
printf("\n%s",*(jedzenie+1)); _____
printf("\n%s",*jedzenie+1); _____
printf("\n%s",*(jedzenie+3)+2); _____
printf("\n%c",*(jedzenie[1])); _____
printf("\n%c",*(jedzenie[2]+3)); _____
printf("\n%c",jedzenie[3][3]); _____
printf("\n%c",jedzenie);** _____

W jaki sposób można wyświetlić poniższe podsłowa?

rozna _____
nikiel _____
NIKIEL _____
f _____

7. Poniższa tabela przedstawia fragment zawartości pamięci komputera:

adres	zawartość
12ff01	123
12ff02	12ff04
12ff03	12ff05
12ff04	12
12ff05	33
12ff06	12ff01

Wiadomo, że zmienna **x** ma adres 12ff02, a zmienna **y** ma adres 12ff03.
Co wyświetli się na ekranie i dlaczego?

printf("%d",*x+*y); _____
printf("%d",*x * *y); _____
printf("%d %d",&x,&y); _____
printf("%d",&x+&y); _____

Jak można wyświetlić poniższe liczby korzystając tylko z **x** i **y** i z operacji na wskaźnikach?

123 _____
12ff01 _____
12 _____