Wielopoziomowe menu w C

Jednym z największych „pożeraczy" czasu i energii programisty, obok uruchomienia programu, jest tak zwany interfejs użytkownika. Mikrokontroler nie potrzebuje do pracy wyświetlacza, klawiatury, diod świecących LED i temu podobnych elementów, o ile nie jest to ściśle związane z operacjami zapisu lub odczytu danych. Wszystkie te drobiazgi są potrzebne nam, ludziom, abyśmy mogli wprowadzać parametry i poznać rezultat wykonania przez mikrokontroler określonego programu.
W artykule przedstawię sposoby tworzenia jedno- i dwupoziomowych menu, wyświetlanych na typowym wyświetlaczu LCD 4 linie po 20 znaków.

Opisane w artykule funkcje wykonałem jako demonstrację sposobu implementacji menu. Nie są one związane z żadnym konkretnym urządzeniem. Do ich uruchomienia użyłem płytki eksperymentalnej własnej konstrukcji, zbliżonej funkcjonalnie do AVR Starter Kit i zawierającej mikrokontroler AT89S8252, programator oraz złącza doprowadzające sygnał do wyprowadzeń mikrokontrolera. Do płytki podłączyłem wyświetlacz LCD (pracuje z interfejsem 4-bitowym) oraz klawiaturę wykonaną jako matrycę 3 wierszy i 4 kolumn (w sumie 12 klawiszy). Port P2, do którego podłączyłem klawiaturę, dodatkowo wyposażyłem w rezystory *pull-up* o wartości 47 kΩ. Użyłem ich ze względu na długie kable łączące klawiaturę z portem mikrokontrolera. W **tab. 1** i na **rys. 1** zawarto opis połączeń elektrycznych.

Odczyt klawiatury

Funkcje odczytujące klawiaturę zadeklarowałem jako zewnętrzne funkcje biblioteczne i umieściłem w module o nazwie KBD3X4. Pierwsza z nich *char KBD_Read(char column)* służy do obsługi klawiatury na poziomie sprzętu, operując na portach mikrokontrolera. Druga - *char Key_Number(void)* - przydziela klawiszom odpowiednie kody.

Funkcja *KBD_Read* w wyniku działania zwraca unikatową dla każdego z wciśniętych klawiszy, odpowiadającą mu wartość lub 0 - jeśli żaden z klawiszy nie jest wciśnięty. Nie pracuje ona poprawnie, jeśli wciśnięto więcej niż jeden klawisz. Zawsze zwracana jest jako pierwsza kombinacja klawiszy z kolumny będącej pierwszą odczytaną. Jako argument wywołania spodziewane jest wyzerowanie bitu odpowiadającego odczytywanej kolumnie. W związku z tym, że kolumna pierwsza klawiatury podłączona jest do P2.0, druga do P2.1 a trzecia do P2.2, odczytowi pierwszej kolumny odpowiada wyzerowanie bitu numer 0 argumentu (argument wywołania np. 0xFE), drugiej bitu numer 1 itd. W pliku nagłówkowym KBD3X4.H za pomocą dyrektywy *#define* zdefiniowano właściwe wartości argumentu wywołania funkcji *KBD_Read* (*Column_1...3*), które później używane są przez funkcję *Key_Number*.

Sposób funkcjonowania *KBD_Read* jest następujący: argument *column* jest zapisywany do portu klawiatury i tym samym linia którejś z kolumn przyjmuje stan niski. Na skutek przyciśnięcia (zwarcia) klawisza linia wiersza - będąca normalnie w stanie wysokim - przejdzie w wymuszony stan niski. Stan ten jest dwukrotnie odczytywany przez mikrokontroler: za drugim razem po 50 milisekundach. Zrobiłem tak, aby upewnić się, że klawisz został naciśnięty celowo i było to zamiarem użytkownika. Jest to również bardzo prosty filtr zakłóceń mogących pojawić się na doprowadzeniu portu.

Aby zapobiec wpływowi na wynik działania funkcji bitów innych niż przyporządkowane wierszom klawiatury, są one maskowane (ustawiane w stan niski) poleceniem *curr &= Row_mask* (iloczyn bitowy). Wartość maski również zdefiniowana została w pliku nagłówkowym. Bit aktywny przyjmuje stan niski. W związku z tym, że łatwiej jest porównywać bity znajdujące się w stanie wysokim, wartość zwracanej zmiennej jest negowana (polecenie *return(~curr)*, „~" to operator dopełnienia jedynkowego).

Funkcja *Key_Number* przyporządkowuje kombinacjom bitów odczytanych z portu mikrokontrolera określone kody. Konstrukcja programu zbliżona jest do budowy klawiatury. Warunki *if* tworzą rodzaj matrycy, na której przecięciach znajdują się kody klawiszy. Wartości *Row_1...4* również predefiniowane są w pliku nagłówkowym. Jedynka na pozycji bitu wiersza odpowiada wciśnięciu klawisza. W przypadku, gdy żaden z klawiszy nie jest naciśnięty, funkcja zwraca wartość 0.

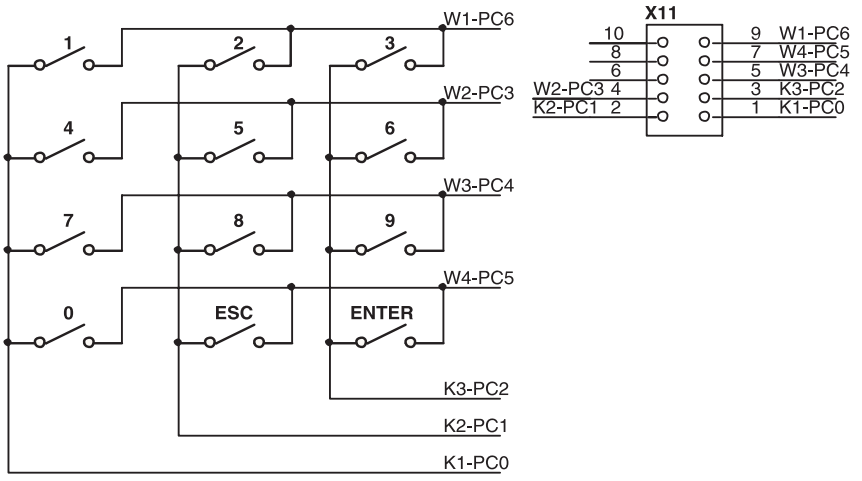
Tab. 1. Przypisanie funkcji wyprowadzeniom mikrokontrolera

Nazwa portu	Oznaczenie	Funkcja
P2.0	PC0	Doprowadzenie pierwszej kolumny matrycy klawiszy
P2.1	PC1	Doprowadzenie drugiej kolumny matrycy klawiszy
P2.2	PC2	Doprowadzenie trzeciej kolumny matrycy klawiszy
P2.3	PC3	Doprowadzenie drugiego wiersza matrycy klawiszy
P2.4	PC4	Doprowadzenie trzeciego wiersza matrycy klawiszy
P2.5	PC5	Doprowadzenie czwartego wiersza matrycy klawiszy
P2.6	PC6	Doprowadzenie pierwszego wiersza matrycy klawiszy
P3.4	PD4	Sygnał RS wyświetlacza LCD
P3.5	PD5	Sygnał READ wyświetlacza LCD
P3.6	PD6	Sygnał ENABLE wyświetlacza LCD
P1.4	PB4	Bit 5 szyny danych wyświetlacza LCD
P1.5	PB5	Bit 6 szyny danych wyświetlacza LCD
P1.6	PB6	Bit 7 szyny danych wyświetlacza LCD
P1.7	PB7	Bit 8 szyny danych wyświetlacza LCD

Menu jednopoziomowe

Najprostszym rodzajem menu jest menu jednopoziomowe. Charakteryzuje się ono tym, że po wybraniu opcji następuje bezpośrednie wywołanie akcji. Menu takie jest łatwe do wykonania, jeśli w całości mieści się na ekranie LCD. Znacznie gorzej jest, gdy ma ono na przykład 8 pozycji, a na ekranie mamy do dyspozycji 2 czy 4 linie. Wówczas musimy nie tylko określić nazwy poszczególnych opcji, ale również sposób wyświetlania menu po przekroczeniu rozmiaru ekranu. Wyliczenie tych wszystkich zmiennych zajmuje dużo pamięci programu oraz czasu mikrokontrolera.

Początek programu (część deklaracyjna zmiennych) zawiera definicję typu o nazwie *menuitem*. Jest to struktura pojedynczego rekordu opisującego linię menu. Rekordy te są następnie zgrupowane w tabeli *menus*, tam też nadawana jest im wartość. Kilku słów omówienia wymaga rola poszczególnych pól rekordu.



Rys. 1. Schemat klawiatury matrycowej 3 (kolumny) x 4 (wiersze) wykorzystywanej w modelu

Pole o nazwie *name* to łańcuch znaków wyświetlany na ekranie i stanowiący nazwę (etykietę) danej pozycji menu. Jest on widziany przez użytkownika dokonującego wyboru opcji. Pole *ptr* to offset dodawany do adresu początku tablicy - wykazu menu. Jego rola polega na wskazaniu miejsca, od którego mają być wyświetlane etykiety menu. I tak dla przykładu *ptr=4* spowoduje, że w linii numer 1 wyświetlacza LCD znajdzie się etykieta menu spod adresu *menus+4*. Kolejne pole - *y0* - to numer linii, w której ma się znajdować znak wyboru w momencie, gdy wskaźnik aktualnej pozycji w tabeli-wykazie wskazuje na rekord związany z tym parametrem. Kombinacja dwóch ostatnich parametrów umożliwia efektywne i czytelne wyświetlanie zawartości menu. Spójrzmy dla przykładu na deklarację menu pochodzącą z prezentowanego programu:

```
//deklaracja tablicy-wykazu
//opcji menu
code menuitem menus[maxoptions] =
{{"opcja 1 . ", 0, 0},
  {"opcja 2 . ", 0, 1},
  {"opcja 3 . ", 0, 2},
  {"opcja 4 . ", 0, 3},
  {"opcja 5 . ", 1, 3},
  {"opcja 6 . ", 2, 3},
  {"opcja 7 . ", 3, 3},
  {"opcja 8 . ", 4, 3}}

//deklaracja wskaźnika do pozycji
//w tablicy-wykazie menu
code menuitem *ptrmenus = &menus;
```

Wskaźnik *ptrmenus* jest tego samego typu, co element tablicy - wykazu. W związku z tym operacje arytmetyczne wykonywane na wskaźniku dodają (lub odejmują) do (od) jego wartości liczbę bajtów identyczną z rozmiarem pojedynczego elementu tablicy. Na przykład wykonanie poleceń:

```
ptrmenus = &menus
ptrmenus = ptrmenus+3  //równoważne
//polecenie to ptrmenus += 3
```

spowoduje przesunięcie wskazań na pozycję 4 w tablicy (pozycje numerowane są od 0), a nie na literę „c" w napisie „opcja 1 .", co odpowiadałoby dodaniu do wartości wskaźnika 3 bajtów. Tyle słowem przypomnienia o arytmetyce wskaźników. Wróćmy do sposobu definiowania menu.

Po uruchomieniu funkcji wyświetlającej, wskaźnik do wykazu ustawiany jest na jego początek (pozycja 0). Następnie pobierane są wartości *menus[0].y0* oraz *menus[0].ptr*. Wartość *ptr* dodawana jest do adresu początku tablicy (w tym przypadku *ptr=0*) i od wyliczonego w ten sposób adresu wyświetlane są na ekranie LCD 4 etykiety menu.

Parametr pobrany z tablicy *y0* to argument funkcji wyświetlającej znaczniki wyboru. W tym przypadku, ponieważ *y0=0*, znaczniki będą wyświetlone na współrzędnych (0,0) oraz (19,0). Podobnie będzie, gdy wskaźnik pokaże pozycję 2, 3 i dalsze. Za każdym razem wartości *y0* i *ptr* określają pozycję znaku wyboru oraz tę etykietę menu, która znajdzie się w pierwszej linii wyświetlacza.

Do wyliczenia wartości wskaźnika do aktualnej pozycji w tabeli-wykazie menu posługuję się zmienną globalną o nazwie *actpos*. Jest to zmienna typu *signed char* (z zakresu od -128...127). Jej wartość ulega zmianie po naciśnięciu klawiszy umownie przyjętych jako kierunek dół (8) lub góra (2). Dodatkowo wartość tej zmiennej porównywana jest z liczbami określającymi jej maksimum i minimum. W przypadku przekroczenia maksimum, zmiennej *actpos* nadawana jest wartość minimum i vice versa: gdy jej wartość jest mniejsza niż wartość minimalna, zmiennej nadawana jest wartość maksimum. Tu jedna uwaga: wartość maksymalną stanowi liczba elementów tablicy *menus*. Jej elementy numerowane są od 0. W związku z tym numer ostatniego elementy tablicy to *rozmiar-1*, a pierwszego to 0. Na skutek tak przyjętej konwencji również i zmiennej *actpos* należy w przypadku dolnej granicy zakresu nadać wartość maksymalnej liczby opcji pomniejszoną o 1. Odpowiada za to polecenie *if (--actpos < 0) actpos=maxoptions-1*. Podobnie jest w przypadku górnej granicy. Wskaźnik adresuje obszar leżący poza tablicą już wówczas, gdy przyjmuje wartość jej rozmiaru. Z mojej praktyki wynika, że mimo faktycznie zachodzącej równości, w takiej sytuacji bezpieczniej jest użyć znaku „większe lub równe". W związku z tym linia programu przyjmuje postać *if (++actpos >= maxoptions) actpos=0*.

Program główny wykonuje czynności związane z obsługą wyświetlacza (inicjalizacja trybu 4-bitowego oraz definicja znaków użytkownika), a następnie wywołuje funkcję obsługi menu. Zwracana przez tę funkcję wartość rozpatrywana jest następnie przez wyrażenie *switch - case*. Oczywiście można użyć instrukcji *if* lub chociażby zbudować tablicę z wykazem wywoływanych funkcji, a zwróconą w wyniku działania wartość potraktować jako offset lub wskaźnik do tejże tablicy.

Menu dwupoziomowe

Rozbudowując menu o dodatkowy poziom, zawsze należy liczyć się z tym, że zajmie ono dużo miejsca

List. 1. Sposób budowania definicji menu

```
code menuitem mainmenus[6] =
{{name = "opcja menu głównego 1",      |-->   code submenu submenu1[3] =
ptr = 0,                               |      {{"podmenu 1.1",0,0,11},
y0 = 0,                                |          {"podmenu 1.2",0,1,12},
nextmenu = &submenu1, ----------------|          {0,0,0,0}}
retval = 0},

{name = "opcja menu głównego 2",       --> code submenu submenu2[7] = {
ptr = 0,                               |      {{"podmenu 2.1",0,0,21},
y0 = 1,                                |          {"podmenu 2.2",0,1,22},
nextmenu = &submenu2, ----------------|          {"podmenu 2.3",0,2,23},
retval = 0},                                    {"podmenu 2.4",0,3,24},
                                                {"podmenu 2.5",1,3,25},
                                                {"podmenu 2.6",2,3,26},
                                                {0,0,0,0}}

{name = "opcja menu głównego 3",
ptr = 0,
y0 = 2,
nextmenu = 0,
retval = 3},
.............
```

w pamięci mikrokontrolera. Sytuacja wygląda zupełnie inaczej, gdy tworzona jest aplikacja dla komputera PC niż gdy dla mikrokontrolera. Ten pierwszy w porównaniu z drugim posiada niewyobrażalne wręcz zasoby do wykorzystania, istne „morze pamięci", przy dodatkowo bardzo dużej szybkości działania. Pewnych nawyków i przyzwyczajeń (o ile takie są) nie da się po prostu w tym momencie wykorzystać, chociaż nowe, pojawiające się na rynku mikrokontrolery imponują wręcz swoimi zasobami sprzętowymi. Zostawmy jednak rozważania i wróćmy do opisu sposobu wykonania menu.

Przy jego budowie posłużyłem się rozwiązaniami z pierwszego przykładu programowania, jednak wyraźnie zostały rozgraniczone menu główne od menu podrzędnego. Rozgraniczają je zarówno funkcje realizujące wyświetlanie etykiet i wybór spomiędzy nich, jak i typy rekordów użytych do deklaracji. Rozpocznijmy analizę od menu głównego. Dla potrzeb budowy tablicy-wykazu opcji menu głównego zadeklarowałem typ o nazwie *menuitem*. Zawiera on pola poznane już wcześniej (*name*, *ptr* i *y0*) i oprócz nich nowe, o nazwach *nextmenu* oraz *retval*.

Pole *nextmenu* to wskaźnik do obszaru wykazu opcji odpowiadającego mu podmenu. W przypadku, gdy następny poziom nie jest dostępny, wskaźnikowi nadawana jest wartość 0. Z tym polem ściśle związana jest zawartość pola *retval*. Zawiera ono liczbę zwracaną przez funkcję w przypadku wyboru opcji. Jak łatwo się domyślić, wartość wpisana do *retval* nie ma żadnego znaczenia w przypadku, gdy wywoływane jest podmenu. Wówczas to, funkcja obsługi menu zwróci wartość zapisaną w polu *retval* właściwą dla danej opcji (etykiety) podmenu.

Wskaźnik *nextmenu* jest typu *submenuitem*. Ten typ zmiennych został utworzony dla potrzeb tablic-wykazów opcji menu drugiego poziomu. Jest on prawie taki sam, jak używany do potrzeb menu głównego. ale oprócz podstawowych pól (etykieta *name*, położenie znaku wyboru *y0*, offset wskaźnika *ptr* oraz wartości zwracanej przez menu *retval*) nie zawiera żadnych wskaźników do następnych wykazów opcji.

Łatwo jest rozbudować podmenu o następne poziomy. Wystarczy użyć podobnych mechanizmów jak dla budowy poziomu 2. Jest w tym jednak drobna trudność: należy w jakiś sposób podać użytkownikowi informację, z którego menu wybierana jest opcja, ot chociażby przez wyświetlenie ścieżki - opisu miejsca, w którym się znajduje. Trudno jest znaleźć to miejsce na ekranie LCD o niewielkich rozmiarach. W praktyce ograniczam więc menu budowanych przez siebie urządzeń co najwyżej do 2...3 poziomów i zazwyczaj nie ma potrzeby tworzenia dalszej hierarchii. Zamiast tego warto jest przemyśleć układ menu i jego opcje. Wróćmy do sposobu tworzenia definicji menu głównego. Na **list. 1** przedstawiono schemat budowy definicji menu głównego oraz odpowiadających mu opcji podmenu.

Tablica *mainmenu* zawiera wykaz definicji etykiet i parametrów menu głównego. Polu wskaźnikowemu o nazwie *nextmenu* nadawana jest wartość adresu - wskazania do wykazu opcji podmenu, jeśli takie istnieje. Kolorami wyróżniono te przypisania. W takim przypadku pole *retval* ma wartość 0. Gdy podmenu nie istnieje (może to być na przykład opcja „Uruchom", „Oblicz" itp.), polu *retval* nadawana jest wartość, którą ma zwrócić funkcja, a polu *nextmenu* należy przypisać wartość 0.

Wymagane jest, aby każda tablica-wykaz etykiet podmenu była zakończony rekordem, w którym co najmniej pole *name* ma wartość 0. Rekord ten jest niezbędny, ponieważ na podstawie jego położenia obliczana jest przez funkcję wyświetlającą podmenu długość wykazu opcji, a tym samym liczba etykiet do wyświetlenia.

Menu główne wyświetlane jest w identyczny sposób jak to było robione w przypadku menu jednopoziomowego. Jego obsługą zajmuje się funkcja *Main_Menu* i jej funkcje usługowe. Jako rezultat zwraca ona liczbę jednobajtową będącą kodem wybranej opcji.

Po naciśnięciu klawisza o kodzie 0x0C umownie funkcjonującego jako ENTER, funkcja sprawdza, czy pole *nextmenu* przyporządkowane bieżącej pozycji zawiera jakąś wartość. Jeśli tak, funkcja wywołuje obsługę podmenu jako parametr, podając zawartość odpowiedniego pola *nextmenu*. W związku z tym, że chciałem napisać funkcję uniwersalną, zdolną do wyświetlenia podmenu o różnych ilościach etykiet, przy wywołaniu funkcja musi obliczyć, jaka jest liczba pozycji wykazu przekazanego jako argument wywołania. Minimalna wartość to 1, maksymalna 127. Koniec wykazu rozpoznawany jest po tym, że pole *name* (etykieta menu) ma wartość znaku o kodzie 0. Wynik obliczeń zapamiętywany jest w zmiennej *maxsubopt*. Ogranicza ona od góry wartość zmiennej globalnej *POS_IN_SUBMENU*, która nie może być większa niż *maxsubopt-1* i jest sprawdzana po każdym naciśnięciu klawisza w górę czy w dół.

Po wejściu do podmenu zaczyna również działać klawisz o kodzie 0x0B pełniący rolę podobną do ESCAPE w komputerze PC. Jego naciśnięcie powoduje, że następuje powrót do menu głównego bez podejmowania jakiejkolwiek akcji i można wówczas dokonać innego wyboru. Wybór zawsze potwierdzany jest przez naciśnięcie klawisza ENTER.

Podobnie jak w poprzednim przykładzie, program główny zawiera inicjalizację wyświetlacza, wywołanie funkcji obsługi menu oraz konstrukcję switch - case w celu rozpoznania wybranej opcji.

Uwagi na temat programu

Funkcje w moich przykładach często posługują się wskaźnikami do struktur. Ich konstrukcja oraz sposób zapisu jest specyficzny i wymaga kilku słów omówienia zwłaszcza dla tych osób, które spotykają się z nimi po raz pierwszy.

Definicja tabeli zawierającej rekordy jest doskonałym mechanizmem zwłaszcza przy tworzeniu baz danych i plików dyskowych. I chociaż w tych przykładach programów nie są używane żadne zbiory znajdujące się na dysku, to jednak tablica rekordów do złudzenia przypomina strukturę bazy danych, a leżący gdzieś w obszarze *code* pamięci mikrokontrolera wykaz - plik dyskowy. Istnieją dwie podstawowe metody odwoływania się do pozycji w takiej strukturze:

- z zastosowaniem indeksów (np. *menus[1]*, *menus[5]* itd.),
- z zastosowaniem wskaźnika (wskaźników).

Ta pierwsza metoda jest bardzo prosta. Ilustruje ją przykład umieszczony na **list. 2**. Indeks to po prostu numer pozycji w tabeli, do którego żąda dostępu aplikacja. Jeśli mamy do czynienia z tabelą zawierającą 10 elementów, to indeks może przyjąć wartości od 0 do 9. Gdy mamy do czynienia z tablicą struktur, dostęp do pola moż-

List. 2. Odwołanie do tablicy rekordów poprzez indeksy

```
typedef struct          //definicja typu - struktury (rekordu) danych
{
   char x;              //struktura zawiera 3 pola typu char: x, y, colour
   char y;
   char colour;
} attr;

data attr points[10];   //deklaracja tablicy zawierającej rekordy danych

void main()
{
   char temp;
   attr pt;

   points[0].x = 1;     //nadanie wartości polu x tablicy wskazywanemu
                        //przez indeks 0
   temp = points[8].y;  //pobranie wartości pola y wskazywanemu przez indeks 8
                        //do zmiennej temp
   pt = point[2];       //przepisanie rekordu danych z tablicy (indeks=2)
                        //do zmiennej pt będącej
}                       //tego samego typu, co element tablicy
```

List. 3. Odwołanie do tablicy rekordów poprzez wskaźnik

```
typedef struct          //definicja typu - struktury (rekordu) danych
{
   char x;              //struktura zawiera 3 pola typu char: x, y, colour
   char y;
   char colour;
} attr;

data attr points[10];          //deklaracja tablicy zawierającej rekordy danych
data attr *pointer = &points;  //deklaracja wskaźnika oraz nadanie mu wartości
                               //początkowej: wskazania na początek tablicy points

void main()
{
   char temp;
   attr pt;

   pointer->x = 1;             //nadanie wartości polu x tablicy wskazywanemu
                               //przez indeks 0
   temp = (pointer+8)->y;      //pobranie wartości pola y wskazywanemu przez
                               //indeks 8 do zmiennej temp
   pt = pointer+2;             //przepisanie rekordu danych z tablicy (indeks=2)
                               //do zmiennej pt będącej
}                              //tego samego typu, co element tablicy
```

na uzyskać podając po kropce jego nazwę: *zmienna = menu[1].ptr*.

Inaczej ma się sytuacja, gdy odwołanie do struktury odbywa się przez wskaźnik. Posłużmy się tym samym przykładem - na **list. 3** umieszczono program zmodyfikowany w taki sposób, aby odwołania do pól następowały przez wskaźnik a nie przez indeks.

Wskaźnik powinien być tego samego typu, co wskazywany obszar danych. W konsekwencji wszelkie operacje arytmetyczne na nim wykonywane powodują jego zmianę o taką liczbę bajtów, jaki jest rozmiar wskazywanego elementu. Ilustracją tej zasady jest poniższy przykład:

```
data char *pointer_1 = &points;
data attr *pointer_2 = &points;
data char temp = 2;
...
...
...
pointer_1++;  //spowoduje przesunię-
   //cie wskazania zmiennej
   //points[0].x na points[0].y
   //(o 1 bajt)
pointer_2++;  //spowoduje przesunię-
   //cie wskaźnika z rekordu
   //points[0] na points[1] i jest
   //równoważne wyrażeniu
   //pointer_1 =
   //pointer_1 + sizezeof(attr)
pointer_1 += 8;  //spowoduje przesu-
   //nięcie wskaźnika o 8 bajtów, to
   //jest na zmienną points[2].y
pointer_2 += 8;  //spowoduje przesu-
   //nięcie wskaźnika o 8 rekordów,
   //to jest na pozycję 9 w tabeli
pointer_1 += temp; //spowoduje
   //przesunięcie wskazania o 2 baj-
   //ty, to jest na zmienną
   //points[0].colour
pointer_2 += temp; //spowoduje
   //przesunięcie wskaźnika o 2 re-
   //kordy, to jest na pozycję
   //3 w tabeli
```

Odwołanie do pola struktury umożliwia operator wyglądający jak strzałka skierowana w prawo, utworzony ze znaku większości oraz odejmowania (->) następującego bezpośrednio po nim. Stosując go, należy pamiętać, że ma bardzo wysoki priorytet. Operacje *(pointer+temp)->x* oraz *temp+pointer->x* nie są równoważne. Jednak bardziej jaskrawy przykład stanowi porównanie sposobu funkcjonowania wyrażeń *++pointer->x* oraz *(++pointer)->x*. W tym pierwszym przypadku wyrażenie zwiększa zmienną *x* o 1, a nie wskaźnik do struktury. W drugim wskaźnik jest zwiększany przed odwołaniem do struktury i na skutek tego następuje przejście do następnego w kolejności elementu tablicy.

Na tej samej zasadzie wyrażenie:

- **pointer->x* udostępnia coś, na co wskazuje *x*,
- **pointer->x++* zwiększa x po udostępnieniu obiektu wskazywanego przez *x*,
- *(*pointer->x)++* zwiększa coś, na co wskazuje *x*,
- **pointer++->x* zwiększa zmienną pointer po udostępnieniu obiektu wskazywanego przez *x*.

Jak wspomniałem wcześniej, wykorzystując mechanizmy użyte do budowy drugiego poziomu menu łatwo jest dodać następne poziomy. Wystarczy, że typ *submenuitem* będzie zawierał wskaźnik do następnego wykazu menu, a funkcja *Sub_Menu* zapamięta i przechowa stan zmiennej *POS_IN_SUBMENU*. Prawdopodobnie będzie również wymagać słowa kluczowego *reentrant* (kompilator RC-51) powodującego, że funkcja może być wywoływana rekurencyjnie. Rozbudowę pozostawiam jednak inwencji własnej Czytelnika.

Jacek Bogusz, AVT
jacek.bogusz@ep.com.pl