Język C i C++. Podstawy

Motto tego artykułu: *Poza funkcjami bibliotecznymi, czyli sam na sam z językiem C.*

Nie zawsze jest możliwe użycie bogatej biblioteki gotowych funkcji istniejących obok języka C. Typowy przykład to programowanie mikrokontrolerów o niewielkich zasobach pamięciowych. To, czy poznane wcześniej printf(), scanf() i inne funkcje są dostępne czy nie, zależy głównie od jakości IDE (środowiska programistycznego), w którym działamy oraz ilości pamięci, jaką ma do dyspozycji mikrokontroler. Warto wiedzieć, jak sobie poradzić bez gotowych rozwiązań, mając do dyspozycji tylko język C oraz funkcję wysyłającą i odbierająca jeden znak z urządzenia. Może to być konsola (klawiatura, ekran tesktowy), wyświetlacz LCD, port szeregowy RS232, modem bezprzewodowy lub inne urządzenie znakowe.

Zagadnienia do opanowania

1) Sprawdź, jakie prototypy funkcji udostępnia plik nagłówkowy <conio.h>
2) Czym się różnią funkcje putc(), puch(), puchar() (i analogicznie getc(), getch(), getchar())? Przeczytaj w dokumentacji i sprawdź w praktyce. Uwaga. Nie w każdym kompilatorze funkcje putch() lub _putch() są dostępne!

```
#include <conio.h>
void wyslij_znak(char c)
{
     _putch(c);
}

void odebrano_znak(char c)
{
}

int main()
{
     char c;
     do {
          c = _getch(); //odczytaj znak z klawiatury
          odebrano_znak(c); //przekaż ten znak funkcji odebrano_znak(…)
     } while(c!=27); //Esc kończy program
     return 0;
}
```

Zaczęliśmy skromnie, od podstawowego terminala znakowego. W świecie mikrokontrolerów jednoukładowych powyższy kod wygląda trochę inaczej i jest zależny od IDE oraz od modelu mikrokontrolera. Przykładowo mógłby wyglądać tak:

```
void wyslij_znak(char c)
{
     _putch(c); //wyślij 1 znak do portu szeregowego
}

void odebrano_znak(char c)
{
     wyslij_znak(c); //odeślij kopię odebranego znaku (ECHO)
}

__interrupt _getch() //specjalna funkcja (reakcja na przerwanie sprzętowe)
{
     odebrano_znak( pobierz_znak_z_portu_szeregowego() );
}

void main()
{
     włącz_port_szeregowy();
     for(;;); //nieskończona pętla, blokująca zakończenie funkcji main()
}
```

Zielony fragment to najważniejsze punkty programu – czyli właśnie funkcja, która potrafi wysłać jeden znak oraz funkcja, która wykonuje się automatycznie za każdym razem, kiedy odebrany zostanie jeden znak. Pozostała część kodu, jaką za moment dodamy, nie jest zależna od możliwości sprzętu i można zadbać o to, aby ten kod zadziałał praktycznie na dowolnym mikrokontrolerze (oczywiście o ile używamy kompilatora C).

Zadania do realizacji

1) W trakcie realizacji tych zadań NIE używaj printf, cprintf, cout << ani innych gotowych funkcji. Masz prawo posługiwać się WYŁĄCZNIE podanymi funkcjami wyslij_znak oraz odebrano_znak.
2) Mając do dyspozycji `void wyslij_znak(char c)`, napisz funkcję wysyłającą napis o prototypie `void wyslij_tekst(char *t)`. Napis przekazujemy jako wskaźnik znakowy.
3) Przygotuj tablicę char odebrane_znaki[100] , w której będą gromadzone odebrane znaki
4) W funkcji `void odebrano_znak(char c)` rozpoznaj pojawienie się kodu klawisza ENTER. Reakcją powinno być wyświetlenie wszystkich odebranych dotychczas znaków w postaci napisu, a następnie wyczyszczenie tablicy znaków.
5) Analogicznie jak została zapisana przykładowa funkcja liczbahex_na_napis, przygotuj własną funkcję liczbadec_na_napis(), która będzie wyświetlać liczby w systemie dziesiętnym.

```
//pomocnicza funkcja, która przekształca liczbę całkowitą
//z zakresu 0..15 na cyfrę szesnastkową (0123456789ABCDEF)
char dec2hex(unsigned char dec)
{
  if(dec >= 0 && dec <= 9) return '0' + dec;
  if(dec >= 10 && dec <= 15) return 'A' - 10 + dec;
  return 'X';
}

//8-bitowa liczbę przekształca na odpowiadające jej 2 cyfry szesnastkowe
//np. 255 zostanie wysłane jako 2 znaki FU, 10 zostałoby wysłane jako 0A
void liczbahex_na_napis(unsigned char liczba)
{
  wyslij_znak(dec2hex((unsigned char)(liczba / 16)));
  wyslij_znak(dec2hex((unsigned char)(liczba % 16)));
}
```

6) Napisz funkcję, która zamienia wszystkie małe łacińskie litery napisu na wielkie.
7) Napisz funkcję porównującą 2 napisy o prototypie `int czy_rowne(char* a, char *b)`. Niech ta funkcja zwraca 0, gdy napisy są różne, oraz 1 gdy są równe.
8) Jak wyżej, ale niech funkcja będzie niewrażliwa na wielkość liter (tzn. ma zwrócić 1 nawet wtedy, gdy porównujemy dowolne z napisów „help”, „HELP”czy „HeLp”).
9) Rozbuduj program o funkcję void wykonaj_polecenie(char *p). Funkcja powinna rozpoznawać kilka poleceń, np. „help”, „exit". Wyniki działania tych poleceń powinny być w postaci wyświetlenia tekstu na urządzeniu znakowym (u nas będzie to okienko tekstowe).