Politechnika Świętokrzyska
Wydział Elektrotechniki, Automatyki i Informatyki
Katedra Elektrotechniki Przemysłowej i Automatyki
Zakład Urządzeń i Systemów Automatyki

Programowanie komputerów 2

Funkcje

Instrukcja laboratoryjna
(wersja robocza)

Paweł Strączyński
2017

Język C wspiera strukturalny paradygmat programowania. Podprogramy w języku C są realizowane z wykorzystaniem funkcji. Funkcja w języku C jest to wydzielona część programu która przetwarza argumenty oraz może zwracać wartość.

Definicja funkcji składa się z nagłówka funkcji oraz treści nazywanej ciałem funkcji.

```
int nazwa_funkcji(int arg1, int arg2) //NAGŁÓWEK FUNKCJI
{
      //CIAŁO FUNKCJI: zmienne lokalne, instrukcje
      return 0;
}
```

Nagłówek funkcji składa się z:

- deklaracji typu jaki funkcja zwraca,
- nazwy funkcji,
- listy argumentów funkcji.

Jeżeli funkcja nic nie zwraca (odpowiednik procedury z języka Pascal) to przy deklaracji zwracanego typu należy użyć typu `void`. Jeżeli funkcja zwraca wartość to wewnątrz jej ciała musi znaleźć się słowo kluczowe `return` wraz z wartością zwracaną przez funkcję. Słowo kluczowe `return` powoduje też zakończenie działania funkcji. Należy zatem pamiętać aby każdy możliwy sposób działania funkcji kończył się określeniem zwracanej wartości. Za nazwą funkcji w nawiasach okrągłych znajduję się lista zawierająca deklarację parametrów pobieranych przez funkcję.

Jeżeli definicja funkcji występuje w kodzie program za miejscem jej wywołania to należy ją wcześniej zadeklarować. **Deklaracja funkcji** polega na zdefiniowaniu jej prototypu składającego się z nagłówka zakończonego średnikiem.

```
int nazwa_funkcji(int arg1, int arg2); //DEKLARACJA FUNKCJI
```

Poniżej przedstawiono przykłady definiowania funkcji.

Listing 1. Przykłady definicji funkcji w języku C

```
float Odwrotna(float foo)
{
      return 1/foo;
}

int Dodaj(int spam, int ham)
{
      int eggs;
```

```
	eggs = spam + ham;
	return eggs;
}

void HelloWorld(void)
{
	printf("Hello World!");
}
```

Używając zmiennych lokalnych należy pamiętać, że są one niszczone po zakończeniu działania funkcji. Aby zmienne te istniały po zakończeniu działania funkcji należy ich deklarację poprzedzić słowem kluczowym `static`. Poniżej zaprezentowano sposób wywołania przedstawionych na listingu powyżej funkcji.

Listing 2. Wywoływanie funkcji

```
int bar,qux;
bar = Odwrotna(float foo);
qux = Dodaj(bar, 5);
HelloWorld();
```

Zadania do wykonania

Zadanie 1

Napisać program weryfikujący poprawność wprowadzonego przez użytkownika numeru PESEL. Program powinien:

- sprawdzać czy podano PESEL o poprawnej ilości cyfr (11) i wyświetlać informację jeśli ilość ta jest niepoprawna,
- sprawdzić poprawność w oparciu o informacje przedstawione poniżej.

Poszczególne funkcjonalności zrealizować w postaci osobnych funkcji.

Wskazówka:

Numer PESEL jest poprawny jeśli wyrażenie przedstawione poniżej dzieli się bez reszty przez 10.

$$1 \cdot a + 3 \cdot b + 7 \cdot c + 9 \cdot d + 1 \cdot e + 3 \cdot f + 7 \cdot g + 9 \cdot h + 1 \cdot i + 3 \cdot j + 1 \cdot k$$

gdzie litery a-k oznaczają kolejne cyfry numeru.

Zadanie 2

Rozbudować program z zadania poprzedniego o funkcję która dla podanego numeru PESEL (jeżeli jest poprawny) rozpoznawał czy należy do kobiety czy mężczyzny.

Wskazówka:

Informacja o płci osoby zawarta jest na 10. cyfrze. Cyfry parzyste oznaczają płeć żeńską natomiast nieparzyste męską.

Zadanie 3

Poniżej przedstawiono fragment noty katalogowej mikrokontrolera AVR Atmega32. Wykorzystując poniższe informacje napisać program który w oparciu o podaną przez użytkownika zawartość rejestru ADMUX mikrokontrolera na ekranie wypisze:

- jakie źródło napięcia odniesienia zostało wybrane (bity REFS1 i REFS0),
- jaki sposób prezentacji wyniku (równanie do lewej/prawej) został ustawiony (bit ADLAR).

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – REFS1:0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 83. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 83. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

- **Bit 5 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conversions. For a complete description of this bit, see "The ADC Data Register – ADCL and ADCH" on page 217.

Rysunek 1. Fragment noty katalogowej mikrokontrolera AVR Atmega32

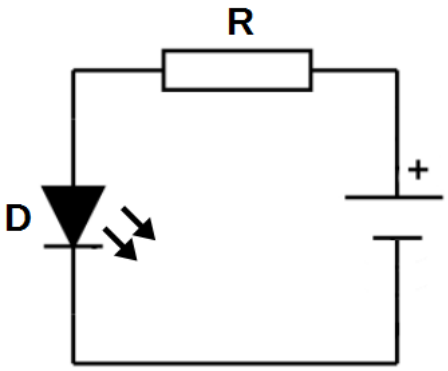
Wykorzystać strukturalny paradygmat programowania.

Zadanie 4

Napisać program który dla przedstawionego na rysunku 2 dobierze wartość rezystora R. Użytkownik powinien mieć możliwość wprowadzenia następujących danych:

- napięcie zasilania układu,
- koloru diody – patrz tabela 1,

Przyjąć że przez diodę powinien przepływać prąd ok. 200mA. Program powinien dobrać najbliższy rezystor z szeregu E6 oraz wyświetlić użytkownikowi faktyczny prąd przepływający przez diodę przy zastosowaniu tego rezystora oraz jego wymaganą moc.



Rysunek 2. Schemat obwodu

Tabela 1. Napięcia przewodzenia dla diod LED

Kolor	**Napięcie przewodzenia**
czerwony	1.2 V
żółty	2.8 V
zielony	3 V
niebieski	3.5 V
biały	3.6 V

Tabela 2. Wartości szeregu E6

10	15	22	33	47	68

Zastosować strukturalny paradygmat programowania.