Politechnika Białostocka
Wydział Elektryczny
Katedra Elektrotechniki, Energoelektroniki i Elektroenergetyki

Instrukcja
do pracowni specjalistycznej z przedmiotu

Programowanie mikrokontrolerów w języku wysokiego poziomu 1

Kod przedmiotu: **TS1F1008**

(studia stacjonarne)

ARDUINO - WSKAŹNIKI I DYNAMICZNY PRZYDZIAŁ PAMIĘCI W JĘZYKU C

Numer ćwiczenia

PMC_13

Autor:
dr inż. Jarosław Forenc

Białystok 2024

Spis treści

Materiały dydaktyczne przeznaczone dla studentów Wydziału Elektrycznego PB.

1. Opis stanowiska

1.1. Stosowana aparatura

Podczas zajęć wykorzystywany jest komputer klasy PC z systemem operacyjnym Microsoft Windows 10/11 oraz platforma Arduino wraz z zestawem czujników.

1.2. Oprogramowanie

Na komputerach zainstalowany jest edytor kodu źródłowego Visual Studio Code 1.92 (lub nowszy) wraz z rozszerzeniem (PlatformIO IDE for VSCode).

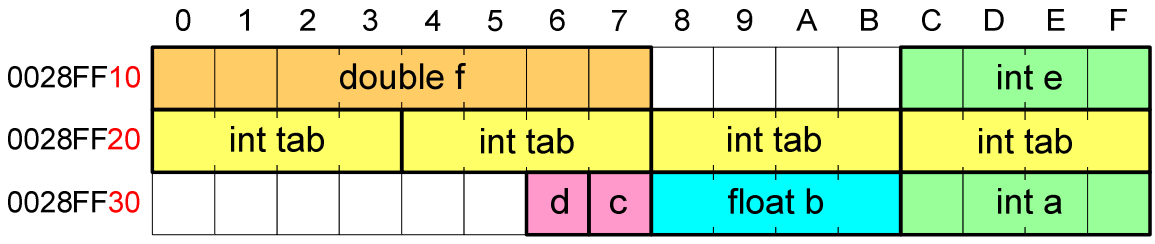
2. Wiadomości teoretyczne

2.1. Wskaźniki

Wskaźnik jest **zmienną** mogącą zawierać adres obszaru pamięci, a w szczególności adres innej zmiennej (obiektu). Załóżmy, że mamy następujące deklaracje zmiennych:

```
int a; float b;
char c, d;
int tab[4];
int e; double f;
```

Wszystkie zmienne są przechowywane w pamięci komputera. W zależności od typu zmiennej zajmują one określoną liczbę bajtów (Rys. 1).



Rys. 1. Zmienne w pamięci komputera

Każda zmienna oprócz nazwy ma także adres. Program nie posługuje się nazwami zmiennych, lecz ich adresami. Na przykład zmienna **a** typu **int** zajmuje 4 bajty i znajduje się pod adresem **0028FF3C** (Rys. 1). Tablica **tab**, przechowująca cztery elementy typu **int**, znajduje się pod adresem **0028FF20** i zajmuje 4 × 4 = 16 bajtów.

Adres określa miejsce w pamięci, ale nie informuje o rozmiarze wskazywanego obiektu. Deklarując wskaźnik (zmienną wskazującą), musimy podać typ obiektu, na jaki on wskazuje. Deklaracja zmiennej wskazującej wygląda tak samo jak deklaracja każdej innej zmiennej, z tą różnicą, że jej nazwa jest poprzedzona symbolem gwiazdki (*):

```
typ *nazwa_zmiennej;
```

lub

```
typ* nazwa_zmiennej;
```

lub

```
typ * nazwa_zmiennej;
```

lub

```
typ*nazwa_zmiennej;
```

Poniższa instrukcja zawiera deklarację zmiennej wskaźnikowej do typu **int**. Oznacza to, że zmienna **ptr** jest typu: **wskaźnik do zmiennej typu int**. Zmienna **ptr** może przechowywać adresy zmiennych **a** i **e** z wcześniejszego przykładu.

```
int *ptr;
```

Do przechowywania adresu zmiennej typu **double** należy zadeklarować zmienną typu: **wskaźnik do zmiennej typu double**.

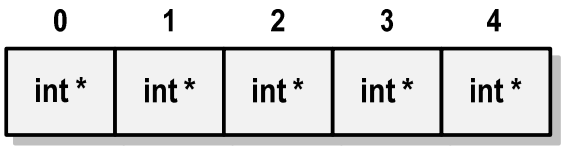
```
double *ptrd;
```

Można konstruować wskaźniki do danych dowolnego typu, łącznie z typami **wskaźnik do wskaźnika do...** . W poniższym przykładzie **wsk** jest typu **wskaźnik do wskaźnika do typu char**.

```
char **wsk;
```

Można deklarować tablice wskaźników. Zmienna **tab_wsk** jest tablicą zawierającą 5 wskaźników do typu **int** (Rys. 2).

```
int *tab_wsk[5];
```



Rys. 2. Tablica zawierająca 5 wskaźników do typu **int**

Natomiast zmienna **wsk_tab** jest wskaźnikiem do 5-elementowej tablicy liczb **int**.

```
int (*wsk_tab)[5];
```

Deklarując wskaźniki, lepiej jest umieszczać znak * przy zmiennej, a nie przy typie:

```
int *ptr1;   /* lepiej */
int* ptr2;   /* gorzej */
```

gdyż trudniej jest pomylić się przy deklaracji dwóch wskaźników:

```
int *p1, *p2;
int* p3, p4;
```

Zmienne **p1**, **p2** i **p3** są wskaźnikami do typu **int**, natomiast zmienna **p4** jest „zwykłą” zmienną typu **int**.

Do przypisania wskaźnikowi wartości (czyli adresu) można zastosować jednoargumentowy operator pobierania adresu **&**:

```
int a = 10;    /* a - zmienna typu int */
int *ptr;      /* ptr - wskaźnik do typu int */
ptr = &a;
```

Poprzez adres zmiennej można „dostać się" do jej wartości używając tzw. **operatora wyłuskania** (**odwołania pośredniego**) - gwiazdki (*****):

```
*ptr = 20;
```

Jeśli zmienna **ptr** przechowuje adres zmiennej **a**, to przypisanie wartości **20** do zmiennej wskazywanej przez **ptr** jest równoważne nadaniu zmiennej **a** wartości **20**:

```
a = 20;
```

Wskaźnik pusty to specjalna wartość, odróżniająca się od wszystkich innych wartości wskaźnikowych, która gwarantuje nierówność z wskaźnikiem do dowolnego obiektu. Do zapisu wskaźnika pustego używa się wyrażenia całkowite o wartości zero (**0**):

```
int *ptr = 0;
```

Często zamiast wartości **0** stosuje się makrodefinicję preprocesora **NULL**, która podczas kompilacji programu jest zamieniana na **0**:

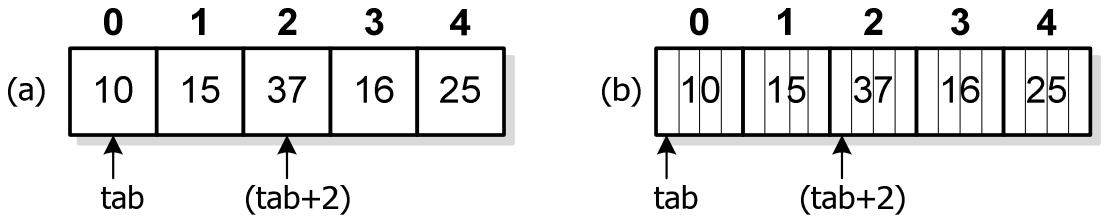
```
int *ptr = NULL;
```

2.2. Związek tablic ze wskaźnikami

W języku C istnieje ścisły związek tablic ze wskaźnikami. Załóżmy, że dana jest następująca deklaracja i inicjalizacja tablicy:

```
int tab[5] = {10,15,37,16,25};
```

Nazwa tablicy jest adresem zerowego elementu tablicy (Rys. 3a). Dokładniej mówiąc, jest to adres pierwszego bajtu zajmowanego przez ten element (Rys. 3b).



Rys. 3. Odwołania do elementów tablicy

Jeśli nazwa tablicy jest adresem zerowego jej elementu, to stawiając symbol * przed nazwą tablicy, możemy „dostać się" do wartości tego elementu. Zatem:

`*tab`	jest równoważne:	`tab[0]`

Podobnie:

`*(tab+1)`
`*(tab+2)`
`...`
	jest równoważne:	`tab[1]`
		`tab[2]`
		`...`

oraz ogólnie:

`*(tab+i)`	jest równoważne:	`tab[i]`

Przed dodaniem liczby całkowitej do wskaźnika, liczb ta jest mnożona przez liczbę bajtów zajmowanych przez wartość wskazywanego typu.

W zapisie ***(tab+i)** nawiasy są konieczne, ponieważ operator * ma wyższy priorytet niż operator **+**.

2.3. Dynamiczny przydział pamięci w języku C

Do dynamicznego przydziału pamięci w języku C stosowane są dwie funkcje: **calloc()** i **malloc()**. Przydział bloków pamięci następuje w obszarze sterty (stosu zmiennych dynamicznych). Przydzieloną dynamicznie pamięć należy zwolnić, wywołując funkcję **free()**.

`calloc()`	Nagłówek: void ***calloc(size_t n, size_t size);**

- funkcja **calloc()** przydziela blok pamięci o rozmiarze **n*size** (mogący pomieścić tablicę **n**-elementów, z których każdy ma rozmiar **size**) i zwraca wskaźnik do tego bloku;
- jeśli nie można przydzielić pamięci, to funkcja zwraca wartość **NULL**;
- przydzielona pamięć jest inicjalizowana zerami (bitowo).

`malloc()`	Nagłówek: void ***malloc(size_t size);**

- funkcja **malloc()** przydziela blok pamięci o rozmiarze określonym przez parametr **size** i zwraca wskaźnik do tego bloku;
- jeśli pamięci nie można przydzielić, to funkcja zwraca wartość **NULL**;
- przydzielona pamięć nie jest inicjalizowana.

Jeśli przydzielony blok pamięci nie jest już potrzebny, to należy zwolnić pamięć, wywołując funkcję **free()**.

`free()`	Nagłówek: void ***free(**void ***ptr);**

- funkcja **free()** zwalnia blok pamięci wskazywany przez parametr **ptr**;
- wartość **ptr** musi być wynikiem wywołania funkcji **calloc()** lub **malloc()**.

W poniższym fragmencie programu przydzielana jest dynamicznie pamięć na jedną zmienną typu **float**, a następnie pamięć ta jest zwalniana.

```
float *wsk;
wsk = (float *) calloc(1,sizeof(float));
free(wsk);
```

Do przydzielenia pamięci zastosowano funkcję **calloc()**. Funkcja ta zwraca wskaźnik do typu **void**, dlatego wartość wskaźnika należy rzutować na właściwy typ (**float** *). Na końcu funkcja **free()** zwalnia pamięć przydzieloną na zmienną.

Kolejny przykład przedstawia dynamiczny przydział pamięci na tablicę jednowymiarową (wektor) liczb całkowitych typu **int**.

Dynamiczny przydział pamięci na tablicę jednowymiarową.

```
#include <Arduino.h>

int *tab, i, n=10;

void setup() {
  Serial.begin(9600);
  tab = (int *) malloc(n*sizeof(int));

  for (i=0; i<n; i++) {
    tab[i] = i*i;
  }

  for (i=0; i<n; i++) {
    Serial.print(tab[i]);
    Serial.println(" ");
  }

  free(tab);
}

void loop() {
}
```

Wynik uruchomienia programu:

```
0
1
4
9
16
25
36
49
64
81
```

Do przydziału pamięci zastosowano funkcję **malloc()**. Funkcja ta zwraca wskaźnik do typu **void**, dlatego wartość wskaźnika należy rzutować na właściwy typ (**int ***). Po przydzieleniu pamięci następuje zapisanie do tablicy kwadratów kolejnych liczb oraz ich wyświetlenie w oknie monitora portu szeregowego. Sposób odwoływania się do elementów tablicy jest identyczny jak w przypadku zwykłej tablicy (np. **tab[i]**). Na koniec funkcja **free()** zwalnia pamięć przydzieloną na tablicę. Przy odwoływaniu się do elementów tablicy można także zastosować operator odwołania pośredniego * (gwiazdka).

```
for (int i=0; i<n; i++) {
   *(tab+i) = i*i;
}
```

3. Przebieg ćwiczenia

Na pracowni specjalistycznej należy wykonać wybrane zadania wskazane przez prowadzącego zajęcia. W różnych grupach mogą być wykonywane różne zadania.

1. Przydziel dynamicznie pamięć na **20**-elementową tablicę liczb całkowitych. Zapisz do tablicy liczby pseudolosowe z zakresu od **1** do **4**. Liczby zapisane w tablicy określają numery diod LED (**1** - **LED1**, **2** - **LED2**, **3** - **LED3**, **4** - **LED4**). Odczytuj kolejne liczby z tablicy i zapalaj odpowiadające im diody LED. Czas

świecenia każdej diody LED powinien wynosić 500 ms. Zwolnij pamięć przydzieloną dynamicznie.

2. Napisz program, którego zadaniem jest wyznaczenie średniej temperatury w sali, w której odbywają się zajęcia. Podczas pracy programu naciśnięcie przycisku **SW1** powinno spowodować zapamiętanie bieżącej temperatury, natomiast naciśnięcie przycisku **SW2** powinno spowodować obliczenie średniej temperatury i wyświetlenie wyniku na wyświetlaczu **OLED**. Wyniki pomiarów należy zapisać w tablicy. Zakładamy, że maksymalna liczba pomiarów wynosi **20**. Przydziel dynamicznie pamięć na tablicę. Stosując napisany program, zmierz temperaturę w różnych punktach sali, a następnie wyświetl średnią temperaturę.

3. Zdefiniuj strukturę składającą się z dwóch pól opisujących częstotliwość i czas trwania dźwięku. Następnie przydziel dynamicznie pamięć na tablicę takich struktur i zapisz do niej dźwięki tworzące wybraną melodię. Po uruchomieniu programu, naciśnięcie przycisku **SW1** powinno spowodować odtworzenie melodii na buzzerze. Pamiętaj o zwolnieniu pamięci.

4. Literatura

[1] Prata S.: Język C. Szkoła programowania. Wydanie VI. Helion, Gliwice, 2016.

[2] Wrotek W.: Arduino od podstaw. Helion, Gliwice, 2023.

[3] Monk S.: Arduino dla początkujących. Podstawy i szkice. Helion, Gliwice, 2019.

[4] Evans M., Noble J., Hochenbaum J.: Arduino w akcji. Helion, Gliwice, 2014.

[5] Kernighan B.W., Ritchie D.M.: Język ANSI C. Programowanie. Wydanie II. Helion, Gliwice, 2010.

[6] https://code.visualstudio.com/ - Visual Studio Code

[7] https://www.arduino.cc/reference/en/ - Arduino Language Reference

5. Pytania kontrolne

1. Wyjaśnij pojęcie wskaźnika i podaj sposób jego deklaracji.
2. Jaki jest związek tablic ze wskaźnikami w języku C?
3. Opisz funkcje do dynamicznego przydzielania i zwalniania pamięci w języku C.

6. Wymagania BHP

Warunkiem przystąpienia do praktycznej realizacji ćwiczenia jest zapoznanie się z instrukcją BHP i instrukcją przeciw pożarową oraz przestrzeganie zasad w nich zawartych.

W trakcie zajęć laboratoryjnych należy przestrzegać następujących zasad.

- Sprawdzić, czy urządzenia dostępne na stanowisku laboratoryjnym są w stanie kompletnym, nie wskazującym na fizyczne uszkodzenie.
- Jeżeli istnieje taka możliwość, należy dostosować warunki stanowiska do własnych potrzeb, ze względu na ergonomię. Monitor komputera ustawić w sposób zapewniający stałą i wygodną obserwację dla wszystkich członków zespołu.
- Sprawdzić prawidłowość połączeń urządzeń.
- Załączenie komputera może nastąpić po wyrażeniu zgody przez prowadzącego.
- W trakcie pracy z komputerem zabronione jest spożywanie posiłków i picie napojów.
- W przypadku zakończenia pracy należy zakończyć sesję przez wydanie polecenia wylogowania. Zamknięcie systemu operacyjnego może się odbywać tylko na wyraźne polecenie prowadzącego.
- Zabronione jest dokonywanie jakichkolwiek przełączeń oraz wymiana elementów składowych stanowiska.

- Zabroniona jest zmiana konfiguracji komputera, w tym systemu operacyjnego i programów użytkowych, która nie wynika z programu zajęć i nie jest wykonywana w porozumieniu z prowadzącym zajęcia.
- W przypadku zaniku napięcia zasilającego należy niezwłocznie wyłączyć wszystkie urządzenia.
- Stwierdzone wszelkie braki w wyposażeniu stanowiska oraz nieprawidłowości w funkcjonowaniu sprzętu należy przekazywać prowadzącemu zajęcia.
- Zabrania się samodzielnego włączania, manipulowania i korzystania z urządzeń nie należących do danego ćwiczenia.
- W przypadku wystąpienia porażenia prądem elektrycznym należy niezwłocznie wyłączyć zasilanie stanowiska. Przed odłączeniem napięcia nie dotykać porażonego.