



Politechnika Białostocka
Wydział Elektryczny
Katedra Elektrotechniki, Energoelektroniki i Elektroenergetyki

Instrukcja
do pracowni specjalistycznej z przedmiotu

Programowanie w języku C

Kod przedmiotu: **TZ1E1004BK**

(studia niestacjonarne)

JĘZYK C - TABLICE JEDNOWYMIAROWE

Numer ćwiczenia

PROGwC_06

Autor:
dr inż. Jarosław Forenc

Białystok 2025

Spis treści

1. Opis stanowiska	3
1.1. Stosowana aparatura	3
1.2. Oprogramowanie	3
2. Wiadomości teoretyczne.....	3
2.1. Tablica elementów	3
2.2. Tablica jednowymiarowa (wektor)	4
2.3. Generowanie pseudolosowe elementów tablicy	11
2.4. Inicjalizacja elementów tablicy	13
3. Przebieg ćwiczenia.....	15
4. Literatura.....	17
5. Pytania kontrolne	18
6. Wymagania BHP	18

Materiały dydaktyczne przeznaczone dla studentów Wydziału Elektrycznego PB.

© Wydział Elektryczny, Politechnika Białostocka, 2025 (wersja 1.0)

Wszelkie prawa zastrzeżone. Żadna część tej publikacji nie może być kopiowana i odtwarzana w jakiegokolwiek formie i przy użyciu jakichkolwiek środków bez zgody posiadacza praw autorskich.

1. Opis stanowiska

1.1. Stosowana aparatura

Podczas zajęć wykorzystywany jest komputer klasy PC z systemem operacyjnym Microsoft Windows 10/11.

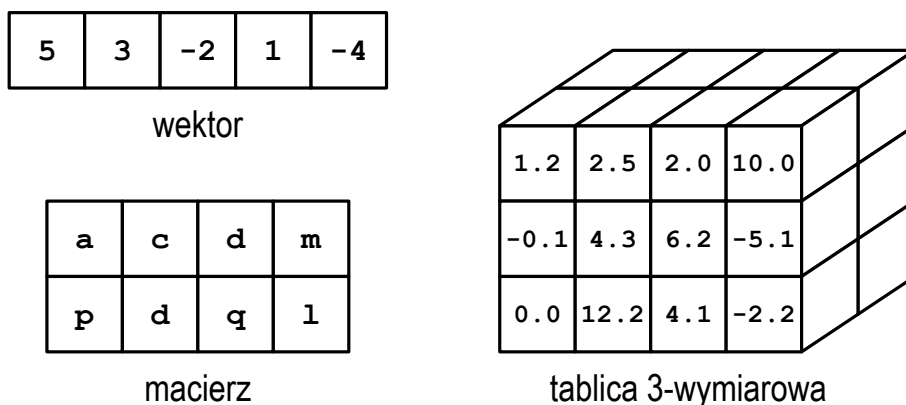
1.2. Oprogramowanie

Na komputerach zainstalowane jest środowisko programistyczne Code::Blocks.

2. Wiadomości teoretyczne

2.1. Tablica elementów

Tablica elementów jest ciągłym obszarem pamięci, w którym te elementy są umieszczone. W tablicy mogą znajdować się elementy tylko jednego typu. Wyróżnia się tablice jednowymiarowe (wektory), dwuwymiarowe (macierze) i tablice o większej liczbie wymiarów (Rys. 1).



Rys. 1. Tablice elementów w języku C

Głównym celem stosowania tablic jest zastąpienie wielu zmiennych tego samego typu jedną tablicą.

2.2. Tablica jednowymiarowa (wektor)

Deklarując tablicę jednowymiarową należy podać: typ elementów, nazwę tablicy i liczbę jej elementów, np.

```
int tab[5];
```

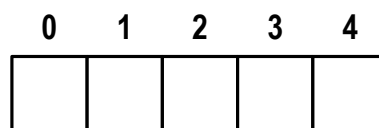
↑
↑
↑
↑
↑

średnik
rozmiar tablicy
nazwa tablicy
typ elementów tablicy

Wyrażenie podane w nawiasach kwadratowych, określające rozmiar tablicy, musi dawać w wyniku dodatnią stałą całkowitoliczbową. Ponadto musi to być wartość znana już w fazie kompilacji (nie może to być zmienna). Jako rozmiar tablicy można podać także nazwę stałej zdefiniowanej dyrektywą preprocesora **#define** lub z użyciem słowa kluczowego **const**.

<pre>int tab[5];</pre>	<pre>#define N 5 int tab[N];</pre>	<pre>const int n = 5; int tab[n];</pre>
------------------------	--	---

Powyższe deklaracje definiują tablicę pięciu elementów typu **int** (Rys. 2). Jest to tablica jednowymiarowa, czyli tzw. **wektor**.

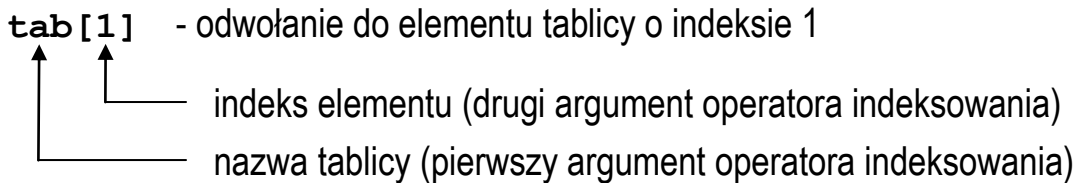


Rys. 2. Wektor 5-elementowy

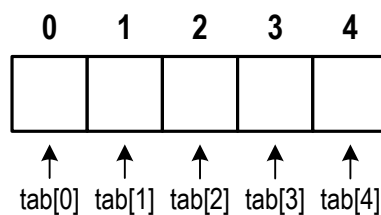
Każdy element tablicy ma swój numer zwany **indeksem**. Element znajdujący się na początku tablicy ma indeks **0** (zero), zaś ostatni element ma indeks **N-1**, gdzie **N** - rozmiar tablicy. Wartość indeksu pokazuje o ile elementów jest dany element odległy od początku tablicy. Nazwa tablicy jest adresem jej zerowego elementu (o indeksie **0**) w pamięci komputera.

Odwołania do elementów tablicy (odczytanie lub zapisanie wartości) wykonuje się za pomocą dwuargumentowego operatora indeksowania `[]`, np.

`tab[1]` - odwołanie do elementu tablicy o indeksie 1



Odwołania do kolejnych elementów tablicy `tab` mają postać pokazaną na Rys. 3.



Rys. 3. Odwołania do elementów tablicy

Zapisanie wartości `5` do elementu tablicy `tab` o indeksie `1` oraz odczytanie tego elementu i przypisanie jego wartości zmiennej o nazwie `x`:

```
tab[1] = 5;  
x = tab[1];
```

Jako indeks może występować:

- stała liczbowa, np. `0`, `1`, `5`;
- nazwa zmiennej przechowującej liczbę całkowitą, np. `i`, `idx`;
- wyrażenie dające w wyniku liczbę całkowitą, np. `i*j + 5`.

Przy odwołaniach do elementów tablicy kompilator nie sprawdza, czy zapis lub odczyt odbywa się w obszarze pamięci przydzielonym na tablicę, np.

```
int tab[5];  
tab[5] = 10;
```

W powyższym fragmencie programu zadeklarowano 5-elementową tablicę o nazwie `tab`. Odwołanie `tab[5]` jest błędne, gdyż nie istnieje element o indeksie `5`.

Kompilator nie zasygnalizuje błędu, tylko w obszarze pamięci za tablicą zapisze wartość **10**.

Operacje na tablicach wykonywane są najczęściej przy wykorzystaniu pętli **for**. Załóżmy, że do wszystkich elementów tablicy **tab** należy zapisać wartość **10**. Kod realizujący taką operację może mieć następującą postać:

```
int tab[5];

tab[0] = 10;
tab[1] = 10;
tab[2] = 10;
tab[3] = 10;
tab[4] = 10;
```

Można to samo zrobić znacznie prościej, stosując pętlę **for**:

```
int tab[5], i;

for (i=0; i<5; i++)
    tab[i] = 10;
```

Zmienna **i** przyjmuje wartości od **0** do **4**, czyli takie same jak kolejne indeksy elementów tablicy.

W poniższym programie przedstawiono najczęściej wykonywane operacje na tablicy jednowymiarowej (wektorze) przechowującej liczby całkowite.

Program wykonujący wybrane operacje na wektorze liczb całkowitych.

```
#include <stdio.h>
#define N 10

int main(void)
{
    int    tab[N], i, j, min, suma = 0, tmp;
    float srednia;
```

```

/* wczytanie elementow tablicy */

for (i=0; i<N; i++)
{
    printf("Podaj liczbe nr %d: ",i+1);
    scanf("%d",&tab[i]);
}

/* wyswietlenie elementow tablicy */

printf("\nElementy tablicy:\n");
for (i=0; i<N; i++)
    printf("%d ",tab[i]);
printf("\n\n");

/* wyswietlenie tablicy w odwrotnej kolejnosci */

printf("Tablica w odwrotnej kolejnosci:\n");
for (i=N-1; i>=0; i--)
    printf("%d ",tab[i]);
printf("\n\n");

/* wyszukanie elementu o najmniejszej wartosci */

min = tab[0];
for (i=1; i<N; i++)
    if (tab[i]<min)
        min = tab[i];
printf("Wartosc elementu najmniejszego: %d\n",min);

/* indeksy elementow o najmniejszej wartosci */

printf("Indeksy elementu najmniejszego: ");
for (i=0; i<N; i++)
    if (tab[i]==min)
        printf("%d ",i);
printf("\n\n");

/* suma i srednia arytmetyczna elementow tablicy */

for (i=0; i<N; i++)
    suma = suma + tab[i];
srednia = (float) suma/N;
printf("Suma: %d, srednia: %f\n\n",suma,srednia);

```

```

/* sortowanie i wyswietlenie elementow tablicy */

for (i=0; i<N-1; i++)
    for (j=i+1; j<N; j++)
        if (tab[i] > tab[j])
            {
                tmp = tab[i];
                tab[i] = tab[j];
                tab[j] = tmp;
            }

printf("Elementy tablicy po sortowaniu:\n");
for (i=0; i<N; i++)
    printf("%d  ", tab[i]);
printf("\n");

return 0;
}

```

Przykładowy wynik uruchomienia programu:

```

Podaj liczbe nr 2: 6
Podaj liczbe nr 3: 4
Podaj liczbe nr 4: 2
Podaj liczbe nr 5: 1
Podaj liczbe nr 6: 7
Podaj liczbe nr 7: 4
Podaj liczbe nr 8: 6
Podaj liczbe nr 9: 3
Podaj liczbe nr 10: 5

```

Elementy tablicy:

```
3 6 4 2 1 7 4 6 3 5
```

Tablica w odwrotnej kolejnosci:

```
5 3 6 4 7 1 2 4 6 3
```

Wartosc elementu najmniejszego: 1

Indeksy elementu najmniejszego: 4

Suma: 41, srednia: 4.100000

Elementy tablicy po sortowaniu:

```
1 2 3 3 4 4 5 6 6 7
```


Rozmiar tablicy określony został przy użyciu dyrektywy preprocesora **#define**.

```
#define N 10
```

Dzięki temu zmiana rozmiaru tablicy będzie wymagała tylko zmiany wartości w dyrektywie **#define**, a nie w każdym innym miejscu programu, gdzie pojawia się ten rozmiar. Dotyczy to zwłaszcza warunków w pętlach **for**.

W programie wykonywane są następujące operacje na tablicy:

- wczytanie elementów tablicy - w pętli **for** wyświetlamy komunikat „**Podaj liczbę nr ...**”, a następnie funkcją **scanf()** wczytujemy liczbę:

```
for (i=0; i<N; i++)
{
    printf("Podaj liczbę nr %d: ", i+1);
    scanf("%d", &tab[i]);
}
```

- wyświetlenie elementów tablicy w jednym wierszu:

```
printf("Elementy wektora:\n");
for (i=0; i<N; i++)
    printf("%d ", tab[i]);
```

- wyświetlenie elementów tablicy w odwrotnej kolejności - zmieniamy wyrażenia w pętli **for**, zmienna **i** będzie przyjmowała wartości od **N-1** (ostatni element tablicy) do **0** (zerowy element tablicy):

```
printf("Tablica w odwrotnej kolejności:\n");
for (i=N-1; i>=0; i--)
    printf("%d ", tab[i]);
```

- wyszukanie elementu o najmniejszej wartości - zakładamy, że zerowy element tablicy jest najmniejszy (**min = tab[0]**); przeglądamy pozostałe elementy tablicy; jeśli kolejny z elementów tablicy (**tab[i]**) jest mniejszy od

dotychczasowego najmniejszego (**min**), to element ten staje się najmniejszym (**min = tab[i]**):

```
min = tab[0];
for (i=1; i<N; i++)
    if (tab[i]<min)
        min = tab[i];
printf("Wartosc elementu najmniejszego: %d\n",min);
```

- wyszukanie indeksów elementu o najmniejszej wartości - przeglądamy tablicę poszukując elementów równych najmniejszemu (**tab[i]==min**); po znalezieniu takiego elementu wyświetlamy jego indeks czyli wartość zmiennej **i**:

```
printf("Indeksy elementu najmniejszego: ");
for (i=0; i<N; i++)
    if (tab[i]==min)
        printf("%d ",i);
printf("\n");
```

- obliczenie sumy i średniej arytmetycznej elementów tablicy - w pętli **for** dodajemy kolejne elementy tablicy do zmiennej **suma** (przed pętlą zmienna ta musi być wyzerowana); następnie obliczamy średnią arytmetyczną dzieląc **sumę** przez ilość elementów (**N**); ponieważ **suma** i **N** są całkowite, to w celu uniknięcia dzielenia liczb całkowitych, wymuszamy zmianę typu zmiennej **suma** na typ **float**: (**float**) **suma**:

```
for (i=0; i<N; i++)
    suma = suma + tab[i];
srednia = (float) suma/N;
printf("Suma: %d, srednia: %f\n",suma,srednia);
```

- sortowanie elementów tablicy w kolejności od najmniejszego do największego
 - pętla zewnętrzna określa indeks elementu (**i**), którego wartość jest porównywana z wartościami pozostałych elementów w tablicy (o indeksach **i+1,i+2,...,N-1**) określanych w pętli wewnętrznej; jeśli kolejność elementów

jest nieprawidłowa (tzn. `tab[i] > tab[j]`), to elementy te zamieniane są miejscami (`tmp = tab[i]; tab[i] = tab[j]; tab[j] = tmp;`):

```
for (i=0; i<N-1; i++)
    for (j=i+1; j<N; j++)
        if (tab[i] > tab[j])
        {
            tmp = tab[i];
            tab[i] = tab[j];
            tab[j] = tmp;
        }
```

Po zakończeniu sortowania elementy tablicy są ponownie wyświetlane:

```
printf("Elementy tablicy po sortowaniu:\n");
for (i=0; i<N; i++)
    printf("%d ", tab[i]);
printf("\n");
```

2.3. Generowanie pseudolosowe elementów tablicy

Elementy tablicy mogą być wygenerowane pseudolosowo, co pokazuje poniższy program.

Generowanie pseudolosowe elementów tablicy.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void)
{
    int tab[8], i;

    srand((unsigned int)time(NULL));
    for (i=0; i<8; i++)
    {
        tab[i] = rand();
        printf("%d ", tab[i]);
    }
}
```

```
    return 0;
}
```

Przykładowy wynik uruchomienia programu:

```
11111 25806 24566 20755 3053 20219 4209 15326
```

Do generowania pseudolosowych liczb zastosowana została funkcja **rand()**:

```
tab[i] = rand();
```

Funkcja ta zwraca pseudolosową liczbę całkowitą z zakresu: **0 ... RAND_MAX (32767)**. Zastosowanie jej w programie wymaga dołączenia pliku nagłówkowego **stdlib.h**. Przed użyciem funkcji **rand()** należy zainicjalizować generator liczb pseudolosowych wywołując funkcję **srand()**:

```
srand( (unsigned int) time(NULL) );
```

której argumentem jest liczba inicjalizująca generator. Aby zapewnić unikalność generowania kolejnych liczb, do funkcji **srand()** przekazywana jest wartość zwracana przez funkcję **time()**. Zastosowanie funkcji **time()** wymaga dołączenia pliku nagłówkowego **time.h**. Wymuszenie konwersji typu **(unsigned int)** likwiduje ostrzeżenie kompilatora Microsoft Visual Studio 2008: **warning C4244: 'argument' : conversion from 'time_t' to 'unsigned int', possible loss of data**, wynikające z niezgodności typu wartości zwracanej przez funkcję **time()** (**time_t**) i typu wartości oczekiwanej przez funkcję **srand()** (**unsigned int**).

Zmiana zakresu generowanych liczb odbywa się poprzez zastosowanie dzielenia modulo. Jeśli chcemy otrzymać liczby całkowite z zakresu **0 ... 10**, to wystarczy wartość zwracaną przez funkcję **rand()** podzielić **modulo 11**:

```
int x;
x = rand() % 11;
```

Pseudolosową liczbę całkowitą z przedziału $\langle a, b \rangle$ otrzymamy używając funkcji **rand()** w następujący sposób:

```
int x;  
x = rand() % (b - a + 1) + a;
```

2.4. Inicjalizacja elementów tablicy

Po zadeklarowaniu tablicy wartości jej elementów są nieokreślone. Inicjalizacja elementów tablicy jest to nadanie wartości elementom od razu przy deklaracji. Inicjalizacja taka polega na umieszczeniu w deklaracji po znaku równości, ujętej w nawiasy klamrowe, listy wartości kolejnych jej elementów, np.

```
int a[3] = {5,7,1};
```

0	1	2
5	7	1

Poszczególne elementy tablicy oddzielone są od siebie przecinkami. Jako kolejne elementy mogą występować liczby lub wyrażenia arytmetyczne. Tablice można inicjalizować **tylko** przy deklaracji.

Jeśli wartości podanych w trakcie inicjalizacji jest mniej niż wynosi rozmiar tablicy, to pozostałe elementy tablicy wypełniane są zerami, np.

```
int a[5] = {5,7,1};
```

0	1	2	3	4
5	7	1	0	0

Jeśli wartości podanych w trakcie inicjalizacji jest więcej niż wynosi rozmiar tablicy, to kompilator zasygnalizuje błąd, np.

```
int a[5] = {5,7,1,3,6,4};
```

Tablica zadeklarowana bez podania rozmiaru, a zainicjalizowana ma liczbę elementów równą ilości inicjatorów, np.

```
int a[] = {2,3,1,4};
```

jest równoważne:

```
int a[4] = {2,3,1,4};
```

Poniższy program używa tablicy do zapamiętania wygenerowanych pseudolosowo, niepowtarzających się liczb całkowitych z zakresu $\langle 1, 49 \rangle$.

Program generujący N niepowtarzających się liczb całkowitych.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N      6
#define ZAKRES 49

int main(void)
{
    int  liczby[N];
    int  i, j, x, powt;

    srand((unsigned int)time(NULL));
    for (i=0; i<N; i++)
    {
        do
        {
            powt = 0;
            x = rand() % ZAKRES + 1;
            for (j=0; j<i; j++)
                if (liczby[j]==x)
                    powt = 1;
        } while (powt==1);
        liczby[i] = x;
    }
    printf("Wylosowane liczby: ");
    for (i=0; i<N; i++)
        printf("%d  ",liczby[i]);
    printf("\n");

    return 0;
}
```

Przykładowy wynik uruchomienia programu:

Wylosowane liczby: 27 30 1 13 22 8

3. Przebieg ćwiczenia

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

1. Zadeklaruj **N**-elementową tablicę liczb całkowitych typu **int** (**N** - stała zadeklarowana dyrektywą preprocesora **#define**). Wykonaj następujące operacje:
 - a) zapisz do tablicy kolejne liczby całkowite **1, 2, 3, ..., N**; wyświetl elementy tablicy w jednym wierszu;
 - b) zapisz do tablicy liczby całkowite **N, N-1, ..., 3, 2, 1**; wyświetl elementy tablicy w jednym wierszu;
 - c) zapisz do tablicy wygenerowane pseudolosowo liczby całkowite z zakresu **<0, 9>**; wyświetl elementy tablicy w jednym wierszu;
 - d) wyświetl w jednym wierszu elementy tablicy o parzystych indeksach;
 - e) wyświetl w jednym wierszu elementy tablicy o nieparzystych indeksach;
 - f) oblicz i wyświetl sumę wszystkich elementów tablicy;
 - g) oblicz i wyświetl średnią arytmetyczną wszystkich elementów tablicy;
 - h) wyświetl wartość największego i najmniejszego elementu tablicy;
 - i) wczytaj z klawiatury liczbę **x**; sprawdź, czy **x** występuje w tablicy; jeśli tak, to wyświetl numer indeksu pierwszego elementu równego **x**;
 - j) wyświetl liczbę wystąpień **x** w tablicy;
 - k) wyświetl liczbę elementów tablicy mniejszych od **x** i liczbę elementów tablicy większych od **x**;
 - l) odwróć kolejność elementów tablicy; wyświetl elementy tablicy w jednym wierszu;

- m) posortuj elementy tablicy w kolejności rosnącej; wyświetl elementy tablicy w jednym wierszu;
- n) posortuj elementy tablicy w kolejności malejącej; wyświetl elementy tablicy w jednym wierszu.
2. Tablica **U** przechowuje wyniki **N**-pomiarów wartości chwilowych napięcia na pewnym dwójniku RLC (przyjmij **N** nie mniejsze niż **15**). Napisz program który:
- a) zapisze do tablicy **U** wartości chwilowe napięcia zgodnie ze wzorem:
- $$U[i] = 10.0f * \sin((i+5.0f)/5.0f); \quad (1)$$
- b) wyświetli na ekranie zapisane wartości chwilowe napięcia (w kolumnie);
- c) obliczy i wyświetli największą, najmniejszą i średnią wartość napięcia;
- d) obliczy i wyświetli liczbę pomiarów, dla których wartość chwilowa napięcia była większa od wartości średniej;
- e) zastąpi w tablicy **U** wszystkie ujemne wartości napięcia wartością zero i ponownie wyświetli wartości elementów tablicy.
3. W pomieszczeniu przeprowadzono pomiar temperatury. Temperaturę mierzono co godzinę (od godz. **0** do **23**). Wyniki pomiarów umieszczono w tablicy **T**. Indeksy elementów tablicy określają jednocześnie godzinę pomiaru. Zakładamy, że wyniki pomiarów nie powtarzają się. Napisz program który:
- a) zapisze do tablicy **T** wartości temperatury zgodnie ze wzorem (dla **i = 0...23**):
- $$T[i] = \sin(i/8.0f-10) * \cos(i/8.0f-10) * 30+10; \quad (2)$$
- b) wyświetli na ekranie zapisane wartości temperatury (w kolumnie);
- c) poda godzinę, o której temperatura była najwyższa oraz godzinę, o której temperatura była najniższa w ciągu całej doby;
- d) obliczy i wyświetli średnią temperaturę w ciągu doby;
- e) obliczy i wyświetli największą różnicę temperatur;
- f) poda informację, czy temperatura w ciągu doby spadła poniżej zera stopni, czy też nie spadła poniżej zera stopni.

4. Dane są dwa **N**-elementowe wektory **A** i **B** zawierające wygenerowane pseudolosowo liczby całkowite z zakresu $\langle 0, 99 \rangle$:
 - a) utwórz wektor **C** zawierający na odpowiedniej pozycji większy z elementów wektorów **A** i **B**,
 - b) utwórz wektor **D** będący sumą wektorów **A** i **B**,
 - c) oblicz i wyświetl iloczyn skalarny wektorów **A** i **B**.Rozmiar wektorów (**N**) zadeklaruj jako stałą (**#define**). Wyświetl wektory **A**, **B**, **C** i **D**.
5. Napisz program, który dla **N**-elementowego wektora liczb całkowitych wygeneruje pseudolosowo elementy z zakresu $\langle 0, 10 \rangle$, wyświetli zawartość wektora oraz obliczy ile razy każda liczba występuje w wektorze.
6. Napisz program, który do **10**-elementowej tablicy liczb całkowitych zapisze wygenerowane pseudolosowo liczby z zakresu $\langle 0, 9 \rangle$. Liczby nie mogą powtarzać się. Wyświetl zawartość tablicy. Podaj ile razy generowano liczbę dla kolejnego elementu tablicy.

4. Literatura

- [1] Prata S.: Język C. Szkoła programowania. Wydanie VI. Helion, Gliwice, 2016.
- [2] Kernighan B.W., Ritchie D.M.: Język ANSI C. Programowanie. Wydanie II. Helion, Gliwice, 2010.
- [3] Deitel P.J., Deitel H.: Język C. Solidna wiedza w praktyce. Wydanie VIII. Helion, Gliwice, 2020.
- [4] Kochan S.G.: Język C. Kompendium wiedzy. Wydanie IV. Helion, Gliwice, 2015.
- [5] King K.N.: Język C. Nowoczesne programowanie. Wydanie II. Helion, Gliwice, 2011.
- [6] <http://www.cplusplus.com/reference/clibrary> - C library - C++ Reference

[7] <https://cpp0x.pl/dokumentacja/standard-C/1> - Standard C

[8] <https://www.codeblocks.org/> - Code::Blocks

5. Pytania kontrolne

1. Omów sposób deklarowania tablic jednowymiarowych (wektorów) w języku C oraz odwoływania się do elementów tych tablic.
2. Wyjaśnij, jak odszukać w tablicy jednowymiarowej element o najmniejszej i element o największej wartości.
3. W jaki sposób w języku C można generować pseudolosowe liczby całkowite i rzeczywiste z określonego zakresu?
4. Opisz inicjalizację elementów tablicy jednowymiarowej.

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.