Informatyka 1

Politechnika Białostocka - Wydział Elektryczny

Elektrotechnika, semestr II, studia niestacjonarne I stopnia

Rok akademicki 2018/2019

Wykład nr 7 (10.05.2019)

dr inż. Jarosław Forenc

Plan wykładu nr 7

- Język C - tablice jednowymiarowe (wektory)
  - deklaracja, odwołania do elementów, inicjalizacja tablicy
  - generator liczb pseudolosowych
  - operacje na wektorze
- System operacyjny
  - definicje systemu operacyjnego
- Zarządzanie procesami
  - definicja procesu, blok kontrolny procesu

Język C - operacje na dużej ilości danych

```c
#include <stdio.h>

int main(void)
{
    double U1, U2, U3, U4, U5;
    double I1, I2, I3, I4, I5;
    double R1, R2, R3, R4, R5;

    U1 = 5.0;
    U2 = 10.0;
    U3 = 15.0;
    U4 = 20.0
    U5 = 25.0;

    I1 = 0.16;
    I2 = 0.21;
    I3 = 0.27;
    I4 = 0.33;
    I5 = 0.36;
```


Język C - operacje na dużej ilości danych

```c
    R1 = U1/I1;
    R2 = U2/I2;
    R3 = U3/I3;
    R4 = U4/I4;
    R5 = U5/I5;

    printf("R1 = %f\n",R1);
    printf("R2 = %f\n",R2);
    printf("R3 = %f\n",R3);
    printf("R4 = %f\n",R4);
    printf("R5 = %f\n",R5);

    return 0;
}
```

```
R1 = 31.250000
R2 = 47.619048
R3 = 55.555556
R4 = 60.606061
R5 = 69.444444
```

Język C - operacje na dużej ilości danych (tablica)

```
R1 = 31.250000
R2 = 47.619048
R3 = 55.555556
R4 = 60.606061
R5 = 69.444444
```

```c
#include <stdio.h>

int main(void)
{
    double U[5] = { 5.0, 10.0, 15.0, 20.0, 25.0 };
    double I[5] = { 0.16, 0.21, 0.27, 0.33, 0.36 };
    double R[5];
    int i;

    for (i=0; i<5; i++)
        R[i] = U[i]/I[i];

    for (i=0; i<5; i++)
        printf("R%d = %f\n",i+1,R[i]);

    return 0;
}
```

	0	1	2	3	4
U	5.0	10.0	15.0	20.0	25.0
I	0.16	0.21	0.27	0.33	0.36
R	31.25	47.62	55.56	60.61	69.44

Język C - tablica elementów

- Tablica - ciągły obszar pamięci, w którym umieszczone są elementy tego samego typu

wektor

5	3	-2	1	-4

macierz

a	c	d	m
p	d	q	l
a	t	x	v

1.2	2.5	2.0	10.0
-0.1	4.3	6.2	-5.1
0.0	12.2	4.1	-2.2

Język C - tablica jednowymiarowa

- Tablica - ciągły obszar pamięci, w którym umieszczone są elementy tego samego typu
- Wektor - tablica jednowymiarowa

5	3	-2	0	-4

- liczby całkowite

3.1	0.2	2.3	-1.3	1.5	1.1	-4.0

- liczby rzeczywiste

a	Z	x	&	M	+

- znaki

Język C - deklaracja tablicy jednowymiarowej

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

rozmiar
nazwa
typ elementów

0	1	2	3	4

indeks

- rozmiar tablicy to wartość:
  - całkowita, dodatnia
  - znana na etapie kompilacji programu
    (stała liczbowa: 5, #define N 5, const int n = 5;)

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

```c
int tab[N];
```

```c
int tab[n];
```

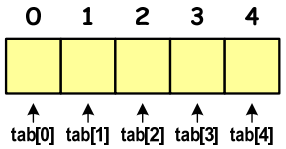
Język C - odwołania do elementów tablicy

[ ] - dwuargumentowy operator indeksowania

```
tab[1];
```

indeks elementu (drugi argument operatora)

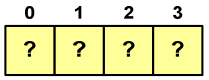
nazwa tablicy (pierwszy argument operatora)



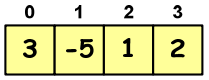
- indeks:
  - stała liczbowa, np. 0, 1, 10
  - nazwa zmiennej, np. i, idx
  - wyrażenie, np. i*j+5

Język C - odwołania do elementów tablicy

```
int tab[4];
```




```
tab[0] = 3;
tab[1] = -5;
tab[2] = 1;
tab[3] = 2;
```



- Każdy element tablicy traktowany jest tak samo jak zmienna typu int

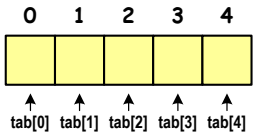
```
printf("%d",tab[0]);
```

```
scanf("%d",&tab[1]);
```


Język C - odwołania do elementów tablicy

- Przy odwołaniach do elementów tablicy kompilator nie sprawdza poprawności indeksów

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



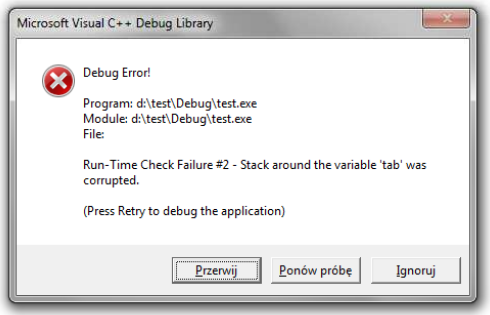
- błąd!!! - nie istnieje element tab[5]

- Kompilator nie zasygnalizuje błędu
- Program wykona operację
- Środowisko programistyczne może zasygnalizować problem

Język C - odwołania do elementów tablicy

- Przy odwołaniach do elementów tablicy kompilator nie sprawdza poprawności indeksów

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



Język C - inicjalizacja tablicy jednowymiarowej

```
int tab[5] = {1,2,3,4,5};
```

0	1	2	3	4
1	2	3	4	5

```
int tab[5] = {1,2,3};
```

0	1	2	3	4
1	2	3	0	0

```
int tab[5] = {1,2,3,4,5,6};
```

- błąd kompilacji

```
int tab[] = {1,2,3,4,5};
```

0	1	2	3	4
1	2	3	4	5

Język C - odwołania do elementów tablicy

- Zapisanie wartości 1 do wszystkich elementów tablicy

```
int tab[5];

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

0	1	2	3	4
1	1	1	1	1

```
int tab[5], i;

for (i=0; i<5; i++)
   tab[i] = 1;
```


Język C - generator liczb pseudolosowych

- rand() - zwraca liczbę pseudolosową - zakres: 0 ... 32767
- srand() - inicjalizuje generator liczb pseudolosowych
- Plik nagłówkowy: stdlib.h (time.h)

```
int x, y;
srand((unsigned int) time(NULL));
x = rand();            // zakres <0,32767>
y = rand() % 100;      // zakres <0,99>
```


Język C - operacje na wektorze

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

#define N 10

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

     /* generowanie elementów tablicy */

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

     for (i=0; i<N; i++)
          tab[i] = rand() % 20;
```

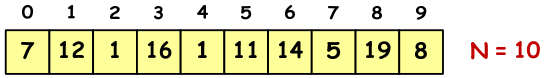
0	1	2	3	4	5	6	7	8	9
7	12	1	16	1	11	14	5	19	8

Język C - operacje na wektorze

```
/* wyświetlenie elementów tablicy */

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

```
Elementy tablicy:
7  12  1  16  1  11  14  5  19  8
```

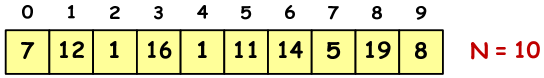


Język C - operacje na wektorze

```
/* wyświetlenie elementów w odwrotnej kolejności */

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

```
Elementy w odwrotnej kolejnosci:
8  19  5  14  11  1  16  1  12  7
```



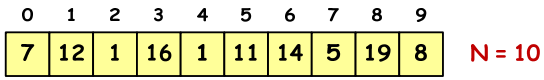
Język C - operacje na wektorze

```
/* wyszukanie elementu o najmniejszej wartości */

int min;

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

```
Wartosc elementu najmniejszego: 1
```

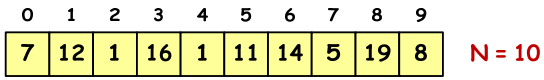


Język C - operacje na wektorze

```
/* indeksy elementów o najmniejszej wartości */

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

```
Indeksy elementu najmniejszego: 2  4
```



Język C - operacje na wektorze

```
/* suma i średnia arytmetyczna elementów tablicy */

int suma = 0;
float srednia;

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

```
Suma: 94, srednia: 9.4
```

0	1	2	3	4	5	6	7	8	9
7	12	1	16	1	11	14	5	19	8

N = 10

Język C - operacje na wektorze

```
/* liczba parzystych elementów tablicy  */

int ile = 0;

for (i=0; i<N; i++)
    if (tab[i]%2==0)
        ile++;
printf("Liczba parzystych elementow: %d\n",ile);
```

```
Liczba parzystych elementow: 4
```

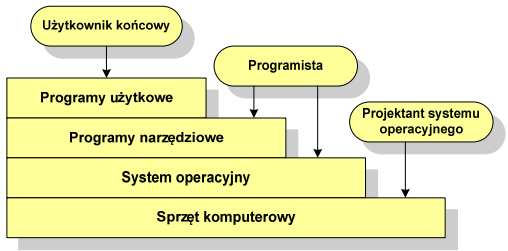
0	1	2	3	4	5	6	7	8	9
7	12	1	16	1	11	14	5	19	8

N = 10

System operacyjny - definicja

- System operacyjny - jest to program sterujący wykonywaniem aplikacji i działający jako interfejs pomiędzy aplikacjami (użytkownikiem) a sprzętem komputerowym

- użytkownik końcowy nie jest zainteresowany sprzętem, interesują go tylko aplikacje (programy użytkowe)
- aplikacje są tworzone przez programistów za pomocą języków programowania



System operacyjny - definicja

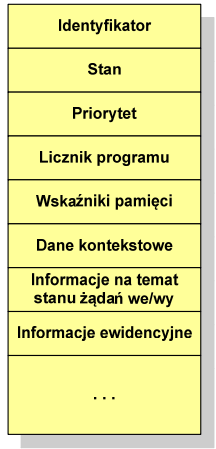
- System operacyjny - administrator zasobów - zarządza i przydziela zasoby systemu komputerowego oraz steruje wykonaniem programu
- zasób systemu - każdy element systemu, który może być przydzielony innej części systemu lub oprogramowaniu aplikacyjnemu
- do zasobów systemu zalicza się:
  - czas procesora
  - pamięć operacyjną
  - urządzenia zewnętrzne

Zarządzanie procesami

- Głównym zadaniem systemu operacyjnego jest zarządzanie procesami
- Definicja procesu:
  - proces - program w trakcie wykonania
  - proces - ciąg wykonań instrukcji wyznaczanych kolejnymi wartościami licznika rozkazów wynikających z wykonywanej procedury (programu)
  - proces - jednostka, którą można przypisać procesorowi i wykonać
- Proces składa się z kilku elementów:
  - kod programu
  - dane potrzebne programowi (zmienne, przestrzeń robocza, bufory)
  - kontekst wykonywanego programu (stan procesu) - dane wewnętrzne, dzięki którym system operacyjny może nadzorować proces i nim sterować

Blok kontrolny procesu

- struktura danych tworzona i zarządzana przez system operacyjny, a opisująca właściwości procesu
- identyfikator - unikatowy numer skojarzony z procesem, dzięki któremu można odróżnić go od innych procesów
- stan procesu: nowy, gotowy, uruchomiony, zablokowany, anulowany
- priorytet - niski, normalny, wysoki, czasu rzeczywistego
- licznik programu - adres kolejnego rozkazu w programie, który ma zostać wykonany
- wskaźniki pamięci - wskaźniki do kodu programu, danych skojarzonych z procesem, dodatkowych bloków pamięci
- dane kontekstowe - dane znajdujące się w rejestrach procesora, gdy proces jest wykonywany
- informacje na temat stanu żądań we-wy - informacje na temat urządzeń we-wy przypisanych do tego procesu



Koniec wykładu nr 7

Dziękuję za uwagę!

(następny wykład: 17.05.2019)