Politechnika Wrocławska

Wstęp do programowania

INP001213Wcl

rok akademicki 2018/19

semestr zimowy

Wykład 4

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

A-1 p. 411B

Politechnika Wrocławska

Plan prezentacji

- Tablice
- Wskaźniki
  - Adresy pamięci
  - Operator adresu (referencji)
  - Operator wskaźnikowy (dereferencji)
- Dynamiczna alokacja pamięci
- Algorytmy sortowania
  - Sortowanie bąbelkowe
  - Sortowanie przez wybieranie

Na podstawie:

- M. M. Sysło, *Algorytmy*
- G. Perry, D. Miller, *Język C Programowanie dla początkujących*

Politechnika Wrocławska

Tablice

- Do przechowywania wielu danych jednego rodzaju (typu) można wykorzystać tablice

- Tablice definiuje się podobnie jak zmienne

```
typ    nazwa[rozmiar];
char   napis[25];
float  oceny[7];
int    punkty[7];
```

Politechnika Wrocławska

Tablice

- Dane do tablic można wprowadzać przy definicji, np.:

```
int liczby[3] = {12, 61, 51};
```

- Można też przypisać wartość wybranemu elementowi, np.:

```
liczby[0] = 12;
liczby[1] = 61;
liczby[2] = 51;
```

Politechnika Wrocławska

Tablice

Start here | tablica_liczb_wczytywanie.c

```c
/*tablica_liczb_wczytywanie.c*/
/*Wstęp do programowania*/

#include <stdio.h>
#define N 7

int main(){
   int i;
   float liczby[N];
   float suma = 0;

   printf("Program wczytuje %d liczb rzeczywistych do tablicy.\n",N);
   printf("Wyswietla tablice, oraz podaje sume elementow i srednia.\n");

```

Politechnika Wrocławska

Tablice

Start here | tablica_liczb_wczytywanie.c

```
15      for(i=0; i<N; i++){
16          printf("Podaj %d. liczbe: ",i);
17          scanf("%f",&liczby[i]);
18      }
19
20      printf("indeks\tliczba\n");
21
22      for(i=0; i<N; i++){
23          printf("%d\t%f\n",i,liczby[i]);
24          suma += liczby[i];
25      }
26
27      printf("Suma = %f,\nsrednia = %f.",suma,suma/N);
28
29      return 0;
30  }
```

Politechnika Wrocławska

Tablice

Start here

```
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
```

```
}
```

```
Program wczytuje 7 liczb rzeczywistych do tablicy.
Wyswietla tablice, oraz podaje sume elementow i srednia.
Podaj 0. liczbe: 2
Podaj 1. liczbe: 1.5
Podaj 2. liczbe: 1
Podaj 3. liczbe: .5
Podaj 4. liczbe: 0
Podaj 5. liczbe: -.5
Podaj 6. liczbe: -1
indeks  liczba
0       2.000000
1       1.500000
2       1.000000
3       0.500000
4       0.000000
5       -0.500000
6       -1.000000
Suma = 3.500000,
srednia = 0.500000.
```

Politechnika Wrocławska

Liczby losowe

- Do generowania liczb losowych można wykorzystać funkcję **`rand()`** z biblioteki **`stdlib`**.
- Funkcja zwraca liczbę całkowitą z przedziału 0, **`RAND_MAX`**, gdzie **`RAND_MAX`** wynosi co najmniej 32 767.

Politechnika Wrocławska

Liczby losowe

Start here | kostka_rand.c

```
1  /*kostka_rand.c*/
2  /*Wstęp do programowania*/
3  /*Program demonstruje
4  użycie funkcji rand().*/
5
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include <ctype.h>
9  #define N 6
10
11 int main(){
12     int kostka;
13     char znak;
14
15     printf("Program symuluje rzuty kostka.\n");
16     printf("Aby przerwac dzialanie wprowadz znak x.\n");
```

Politechnika Wrocławska

Liczby losowe

Start here | kostka_rand.c

```
17
18      do{
19          kostka = rand() % N + 1;
20          //reszta z dzielenia przez N to liczba od 0 do N-1
21          printf("Na kostce wypadlo %d.\n", kostka);
22          scanf(" %c",&znak);
23      }while(toupper(znak) != 'X');
24
25      return 0;
26  }
```

Politechnika Wrocławska

Liczby losowe

- Generator liczb pseudolosowych działa w sposób deterministyczny.
- W celu zmiany ciągu generowanych liczb należy zmienić ziarno generatora.
- W tym celu wywołać należy funkcję **`srand()`** przekazując do niej liczbę całkowitą.
- Przekazywaną liczbą może być czas wywołania funkcji zwracany przez funkcję **`time()`** z biblioteki **`time`**.

Politechnika Wrocławska

Liczby losowe

Start here | kostka_rand.c | **kostka_srand.c**

```c
/*kostka_srand.c*/
/*Wstęp do programowania*/
/*Program demonstruje
użycie funkcji rand() i srand().*/

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <time.h>
#define N 6

int main(){
    int kostka;
    char znak;
    time_t t;

    printf("Program symuluje rzuty kostka.\n");
    printf("Aby przerwac dzialanie wprowadz znak x.\n");
```

Politechnika Wrocławska

Liczby losowe

Start here | kostka_rand.c | **kostka_srand.c**

```
19
20      srand(time(&t)); //funkcja srand() zmienia ziarno generatora
21      //srand(time(NULL)); - alternatywne wywołanie
22
23      do{
24          kostka = rand() % N + 1;
25          printf("Na kostce wypadlo %d.\n", kostka);
26          scanf(" %c",&znak);
27      }while(toupper(znak) != 'X');
28
29      return 0;
30  }
```

Tablice

Podsumowanie

- Tablice służą do przechowywania danych tego samego typu.
- Do poszczególnych elementów można odwoływać się za pomocą indeksów.
- Za pomocą pętli for można przeglądać elementy tablicy.
- Pamiętaj o ograniczonym rozmiarze tablicy.
- Nie używaj tablicy przed inicjalizacją.
- Do generowania losowych danych można użyć funkcję **`rand()`** z **`stdlib`**.

Politechnika Wrocławska

Wskaźniki

Adresy pamięci

- Pamięć komputera podzielona jest na komórki. Każda komórka pamięci ma swój unikatowy adres.

- Deklaracja zmiennej w programie powoduje przydzielenie odpowiedniej pamięci.
  W programie posługujemy się nazwą zmiennej, która zastępuje adres pamięci.

Politechnika Wrocławska

Wskaźniki

Operatory

Operator	Opis
&	Operator adresu
*	Operator dereferencji

Politechnika Wrocławska

Wskaźniki

Definiowanie zmiennych wskaźnikowych

```
int indeks;
float liczba;

int * wskIndeks;
float * wskLiczba;
```

- Nazwy zmiennych wskaźnikowych nie muszą mieć specjalnego formatu.

Wskaźniki

Operator adresu (referencji)

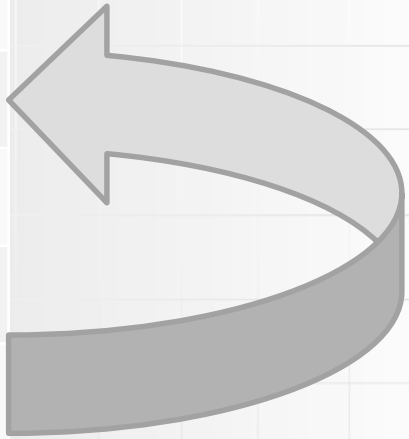
```
int wiek = 19; /*tworzy zmienną
i przypisuje jej wartość 19*/

int * wskWiek = &wiek; /*tworzy
wskaźnik i przypisuje mu adres
zmiennej wiek, ustawia wskaźnik*/
```

Wskaźniki

Operator adresu (referencji)

```
int wiek = 19;
int * wskWiek = &wiek;
```

Adres	Pamięć	Nazwa zmiennej
…	…	
…	…	
18826	19	wiek
…	…	
…	…	
20886	18826	wskWiek
…	…	
…	…	



Politechnika Wrocławska

Wskaźniki

Operator adresu (referencji)

```
int wiek;
int * wskWiek;
wiek = 19;
wskWiek = &wiek;
```

Wskaźniki

Operator wskaźnikowy (dereferencji)

```
wiek = 25;
*wskWiek = 25; /*przypisuje wartość 25
zmiennej wskazywanej przez wskWiek*/
```

- Operatora wskaźnikowego można użyć w instrukcji przypisania

Wskaźniki

Operator wskaźnikowy (dereferencji)

- Podobnie można wykorzystać operator wskaźnikowy, aby odczytać wartość zmiennej

```
printf("Wiek: %d.\n", wiek);
printf("Wiek: %d.\n", *wskWiek);
```

Wskaźniki

Przykład

Start here | wskazniki.c

```
1  /*wskazniki.c*/
2  /*Wstęp do programowania*/
3  /*Program demonstruje
4  dostęp do zmiennych
5  przy użyciu wskaźników.*/
6
7  #include <stdio.h>
8
9  int main(){
10     char znak;
11     char* wskZnak;
12     int liczba;
13     int* wskLiczba;
```

Wskaźniki

Przykład

Start here | wskazniki.c

```
14
15      /*Program prosi o podanie znaku,
16      a następnie go wypisuje.
17      Program prosi o podanie liczby całkowitej,
18      a następnie ją wypisuje.*/
19
20      printf("Podaj znak: ");
21      scanf(" %c",&znak);
22      printf("Podany znak to: %c.\n",znak);
23      printf("Podaj liczbe calkowita: ");
24      scanf("%d",&liczba);
25      printf("Podana liczba to: %d.\n",liczba);
```

Wskaźniki

Przykład

Start here | wskazniki.c

```
26
27      /*Program powtarza działanie
28      z wykorzystaniem wskaźników.*/
29      wskZnak = &znak;
30      wskLiczba = &liczba;
31      printf("Podaj znak: ");
32      scanf(" %c",wskZnak);
33      printf("Podany znak to: %c.\n",*wskZnak);
34      printf("Podaj liczbe calkowita: ");
35      scanf("%d",wskLiczba);
36      printf("Podana liczba to: %d.\n",*wskLiczba);
37      return 0;
38  }
```

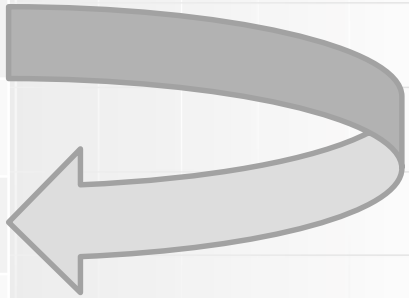
Wskaźniki i tablice

Tablice jako stałe wskaźnikowe

```
int tab[5] = {10, 20, 30, 40, 50};
```

Wartość wskaźnika tab jest stała.

Adres	Pamięć	
32054	46204	tab
…	…	
46204	10	tab[0]
46208	20	tab[1]
46212	30	tab[2]
46216	40	tab[3]
46220	50	tab[4]
…	…	



Wskaźniki i tablice

Przykład

Start here | wskazniki_i_tablice.c

```
1  /*wskazniki_i_tablice.c*/
2  /*Wstęp do programowania*/
3  /*Program demonstruje
4  dostęp do tablicy
5  przy użyciu wskaźników.*/
6
7  #include <stdio.h>
8
9  int main(){
10    int tab[5] = {10, 20, 30, 40, 50};
```

Wskaźniki i tablice

Przykład

Start here | wskazniki_i_tablice.c

```
11
12      //wypisanie początkowego elementu tablicy
13      printf("tab[0]    = %d\n", tab[0]);
14
15      //wypisanie adresu tablicy (jej początku)
16      printf("tab       = %d\n", tab);
17
18      //wypisanie początkowego elementu tablicy
19      printf("*tab      = %d\n", *tab);
20
21      //wypisanie elementu tablicy o indeksie 2
22      printf("tab[2]    = %d\n", tab[2]);
23      printf("*(tab+2)  = %d\n", *(tab+2));
24
25      //wypisanie adresu elementu tablicy o indeksie 2
26      printf("tab+2     = %d\n", tab+2);
27
28      return 0;
29  }
```

Wskaźniki i tablice

Przykład

Start here | wskazniki_i_tablice.c

```
11
12      //wypisanie początkowego elementu tablicy
13      printf("tab[0]    = %d\n", tab[0]);
14
        resu tablicy (jej początku)
                  = %d\n", tab);

        czątkowego elementu tablicy
                  = %d\n", *tab);

        ementu tablicy o indeksie 2
                  = %d\n", tab[2]);
        2)        = %d\n", *(tab+2));

25      //wypisanie adresu elementu tablicy o indeksie 2
26      printf("tab+2     = %d\n", tab+2);
27
28      return 0;
29  }
```

```
tab[0]    = 10
tab       = 6356732
*tab      = 10
tab[2]    = 30
*(tab+2)  = 30
tab+2     = 6356740
```

Dynamiczna alokacja pamięci

Funkcja `malloc()`

```
float *tab;
tab = malloc(10 * sizeof(*tab));
```

- W powyższym wywołaniu funkcja **`malloc()`** alokuje pamięć na 10 liczb typu **`float`**, następnie wskaźnik **`tab`** jest ustawiany na pozyskaną pamięć.
- Funkcja **`malloc()`** pobiera jeden argument – liczbę bajtów, która ma zaalokować.
- Liczbę bajtów dla danego typu zmiennej można odczytać operatorem **`sizeof`**.

Politechnika Wrocławska

Dynamiczna alokacja pamięci

Funkcja `free()`

- Gdy pamięć pozyskana funkcją **`malloc()`** jest nie potrzebna należy ją zwolnić funkcją **`free()`**.

```
free(tab);
```

- Funkcje **`malloc()`** i **`free()`** znajdują się w bibliotece **`stdlib`**.

Dynamiczna alokacja pamięci

Przykład

Start here | wskazniki_malloc_free.c

```
1  /*wskazniki_malloc_free.c*/
2  /*Wstęp do programowania*/
3  /*Program demonstruje
4  dynamiczną alokację pamięci.*/
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  int main(){
10    int i, n;
11    float *tab;
12
13    printf("Program wczytuje n liczb rzeczywistych\
14   i zapisuje je w tablicy.\n");
15    printf("Podaj n: ");
16    scanf("%d",&n);
17
```

Dynamiczna alokacja pamięci

Przykład

Start here | wskazniki_malloc_free.c

```
17
18      tab = malloc(n*sizeof(*tab));
19
20      for(i=0; i<n; i++){
21          printf("Podaj liczbe: ");
22          scanf("%f",&tab[i]);
23          //scanf("%f",tab+i);
24      }
25
26      for(i=0; i<n; i++){
27          printf("%d\t%f\n",i,tab[i]);
28          //printf("%d\t%f\n",i,*(tab+i));
29      }
30
31      free(tab);
32
33      return 0;
34  }
```

Politechnika Wrocławska

Wskaźniki

Podsumowanie (1)

- Operator **&** służy do pobrania adresu zmiennej.
- Operator ***** służy do definiowania wskaźników oraz do dereferencji zmiennych wskaźnikowych.
- Do elementów tablic można odwoływać się z wykorzystaniem indeksów lub operatorów wskaźnikowych.

Politechnika Wrocławska

Wskaźniki

Podsumowanie (2)

- Funkcje `malloc()` i `free()` służą do dynamicznego alokowania i zwalniania pamięci.
- Funkcja `malloc()` pobiera liczbę bajtów.
- Do obliczenia liczby bajtów przydatne jest wykorzystanie operatora `sizeof`.
- Nie zapomnij o zwalnianiu pamięci.

Politechnika Wrocławska

Sortowanie bąbelkowe

- Jeżeli ciąg nie jest uporządkowany, to istnieją dwa sąsiednie elementy, które są w złej kolejności

Politechnika Wrocławska

Sortowanie bąbelkowe

3	5	1	2	8	4	7	6

3	1	5	2	8	4	7	6

3	1	2	5	8	4	7	6

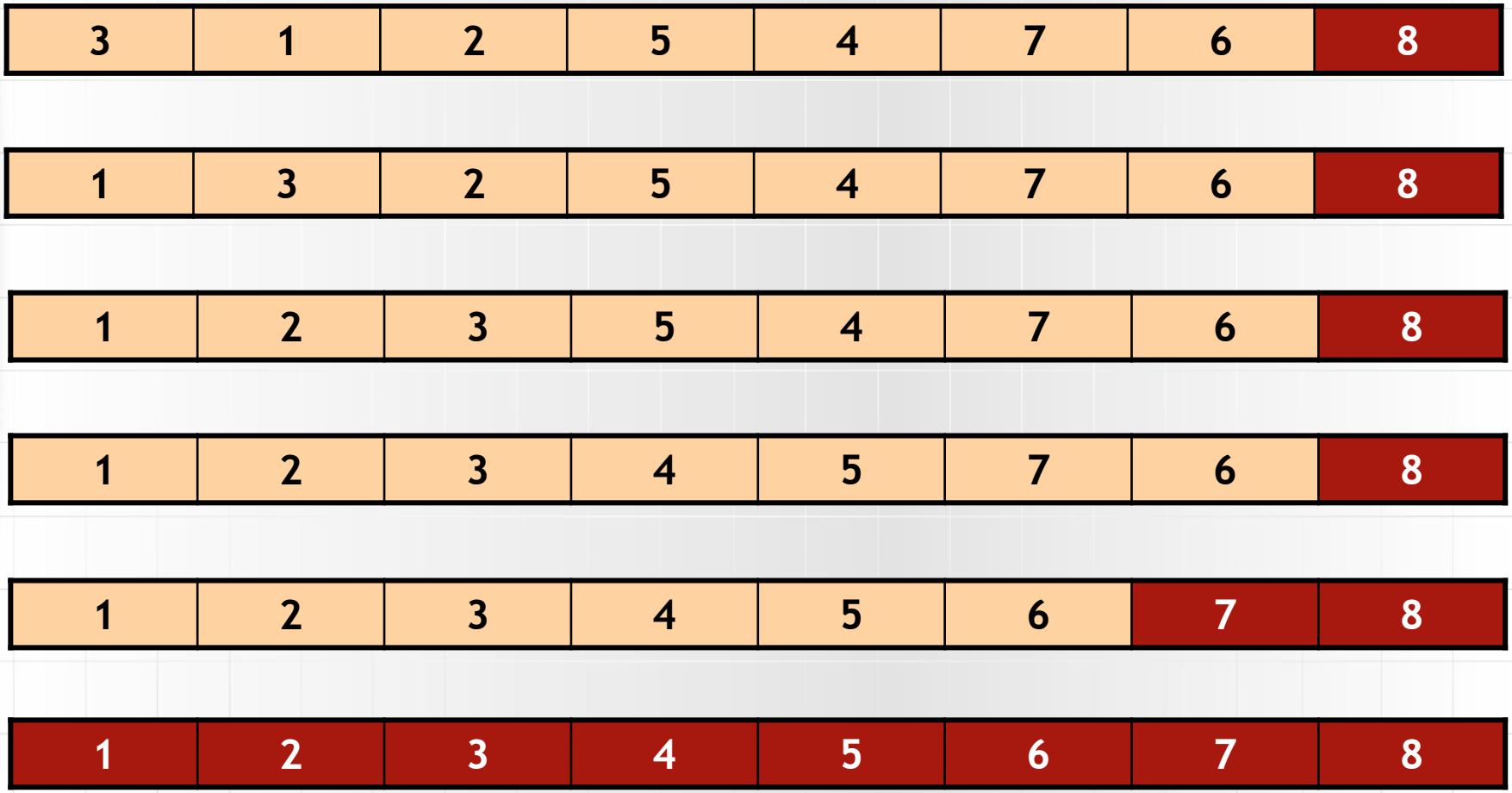
3	1	2	5	4	8	7	6

3	1	2	5	4	7	8	6

3	1	2	5	4	7	6	8

Politechnika Wrocławska

Sortowanie bąbelkowe



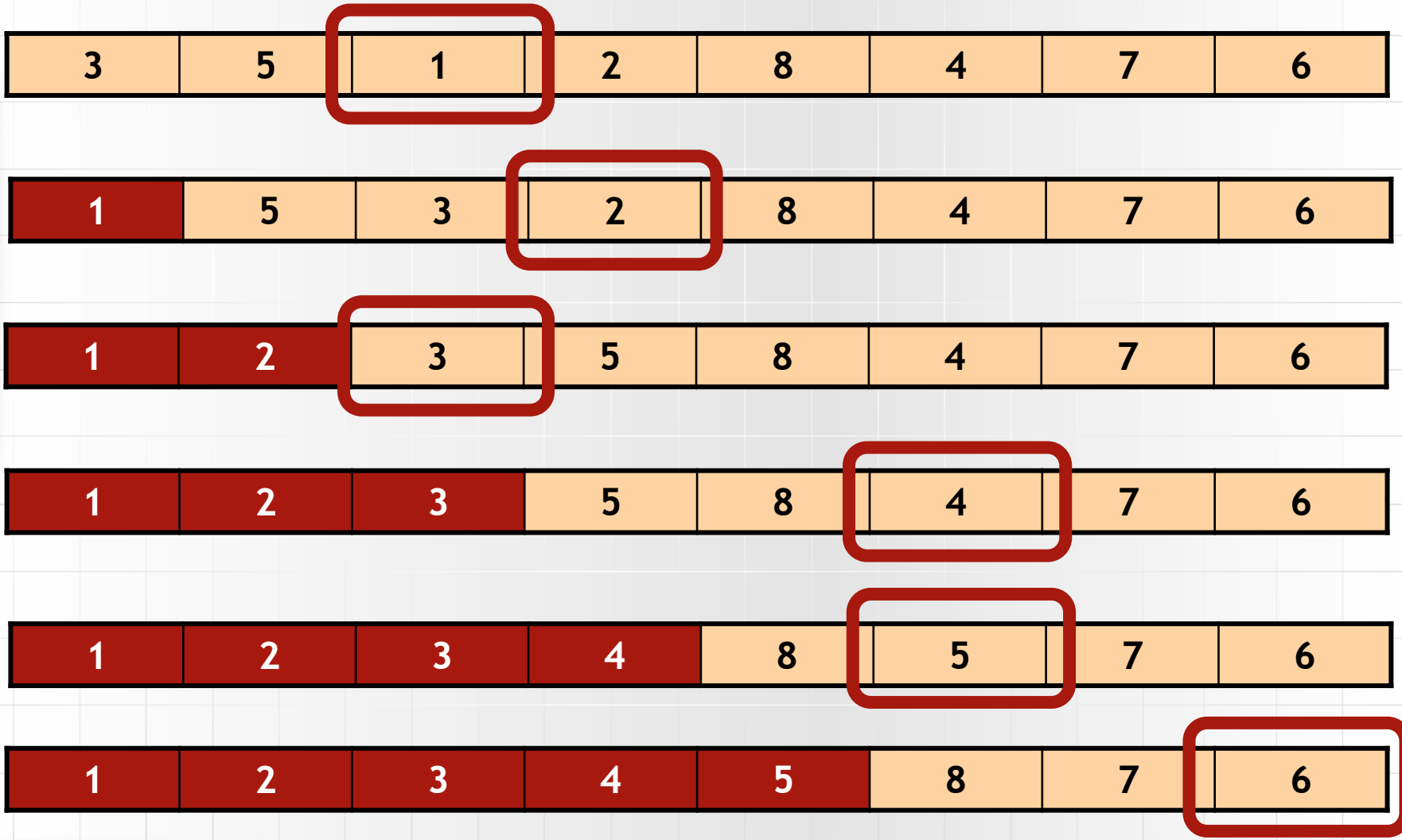
Politechnika Wrocławska

Sortowanie przez wybór

- Wyszukujemy pozycję najmniejszego elementu i przestawiamy go na właściwą pozycję
- Następnie kontynuujemy dla pozostałej części tablicy

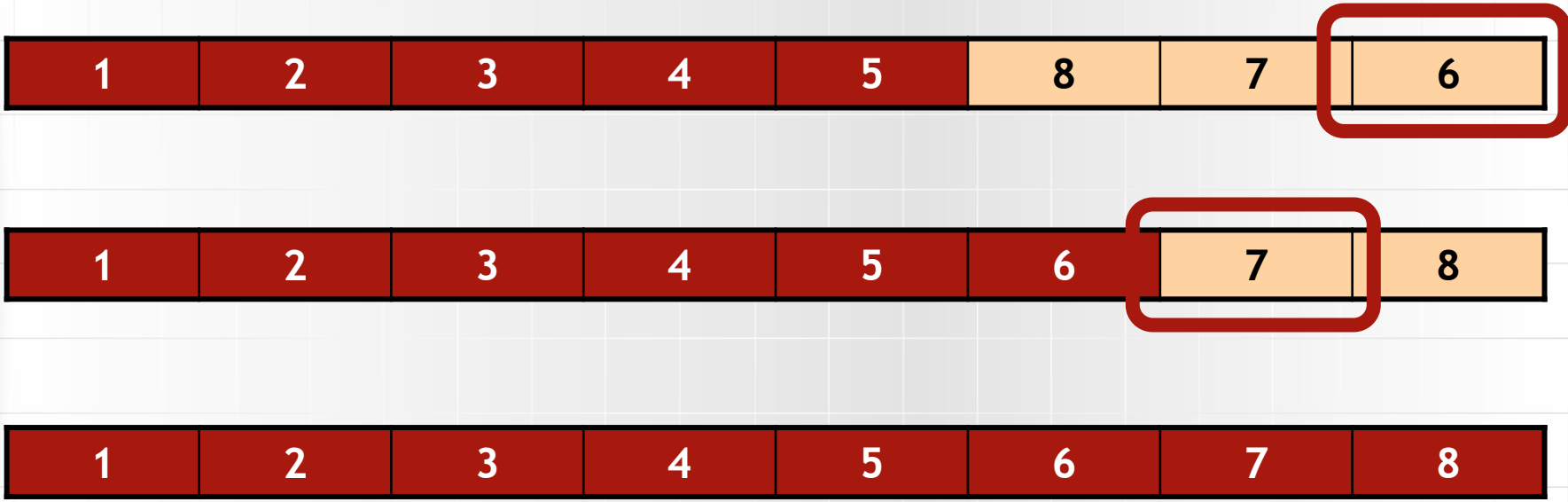
Politechnika Wrocławska

Sortowanie przez wybór



Politechnika Wrocławska

Sortowanie przez wybór



Politechnika Wrocławska

Podsumowanie

- Tablice
- Wskaźniki
  - Adresy pamięci
  - Operator adresu (referencji)
  - Operator wskaźnikowy (dereferencji)
- Dynamiczna alokacja pamięci
- Algorytmy sortowania
  - Sortowanie bąbelkowe
  - Sortowanie przez wybieranie