Algorytmika 2.

(1) Pojęcie tablicy i wybrane algorytmy

a. Wczytanie tablicy Tab[N] (Tab[0], … , Tab[N-1])

```
int Tab[N];
int i;
for(i=0; i<N; i=i+1){
      cout << "Podaj " << i <<"-ty element tablicy: " ;
      cin >> Tab[i];
}
```

b. Wypisywanie tablicy Tab[N]

```
int Tab[N];
int i;
for(i=0; i<N; i=i+1)
    cout << "Tab[" << i <<"]= "<<Tab[i]<<endl ;
```

c. Algorytm obliczający max i min w tablicy Tab[N]

```
int Tab[N];
int max, min, i; max=min=Tab[0];
i=1;
while(i<N) {
  if(Tab[i]>max) max=Tab[i];
  else
    if(Tab[i]<min) min=Tab[i];
  i=i+1;
}
cout « "Min= "« min « endl;
cout « "Max= "« max « endl;
```

d. Algorytm zwracający pierwszą pozycję zadanego elementu x w tablicy lub N gdy $x \notin \{Tab[0],\ldots,Tab[N-l]\}$

```
int Tab[N]; int  i; int  x;
i=0;
while(i<N && x!=Tab[i]) i=i+1;
    // i==N - brak x w tablicy lub x == Tab[i]
if(i==N) cout << "Brak szukanego elementu" << endl;
else cout << "Szukany element jest na pozycji: " <<i<<endl;
```

e. wersja z wartownikiem

```
int Tab[N+1];
int i; int x;
i=0; Tab[N]=x; // wartownik
while( x != Tab[i]) i=i+1;
```

```
// i==N lub x==Tab[i]
if(i==N) cout << "Brak szukanego elementu" << endl;
else cout << "Szukany element jest na pozycji: " <<i<<endl;
```

Zadania.

1. Napisz algorytm sprawdzający porządek w tablicy Tab[N]
2. Napisz algorytm usuwający z tablicy zadany element
3. Napisz algorytm, który wstawia do tablicy uporządkowanej zadany element zachowując porządek tablicy po wstawieniu.
4. Napisz algorytm usuwający duplikaty z tablicy Tab[N]