ISTOTNE UWAGI DO PONIŻSZYCH ZADAŃ:

- Nie należy stosować zmiennych globalnych;
- Kod winien być pisany przejrzyście z uwzględnieniem wcięć podkreślających poziom polecenia;
- Każdy utworzony plik nagłówkowy we wszystkich zadaniach winien zawierać stosowne dyrektywy kompilacji
 warunkowej. Stała symboliczna powinna mieć z góry ustaloną nazwę, adekwatną do nazwy pliku
 nagłówkowego przykładowo dla sortint.h jej nazwa: SORTINT H
- Koniecznie należy stosować dokładnie te nazwy, które podano w treści zadań.
- Począwszy od zadania 18 wywołanie programu bez parametrów spowoduje wyświetlenie składni wywołania danego programu.

Zad. 18.

Po zapoznaniu się z poleceniem **ar** utwórz bibliotekę zawierającą dwie funkcje sortujące (jednowymiarowe) tablice liczb całkowitych. Funkcje winny stanowić implementację dwóch dowolnych i różnych algorytmów sortujących.

Następnie napisz krótki program testujący (zad18.c).

Jeśli użytkownik nie poda żadnego parametru przy wywołaniu programu, to powinien zobaczyć tekst stanowiący instrukcję uruchomienia. Przykład przedstawiono w ramce poniżej.

```
./zad18
usage: ./zad18 -nvalue

Possible values:
bubble - for bubble sort
select - for select sort
```

W programie testującym należy pobierać od użytkownika opcję, po której jako parametr nazwę danego algorytmu. Przykład poprawnych wywołań poniżej:

```
./zad18 -n bubble ./zad18 -nbubble
```

Uwaga:

- Program powinien sprawdzać nie tylko właściwą liczbę argumentów wywołania, ale i użytą opcję oraz jej wartość. Jeśli z któregoś powodu nie można będzie kontynuować, to powinien poinformować o tym użytkownika (tak jak w przykładzie w ramce) i zakończyć działanie.
- W programie testującym użyj z góry zdefiniowaną tablicę statyczną zawierającą konkretne wartości liczbowe.
- Definicja tablicy, jej wypisanie itp. powinno mieć miejsce dopiero po sprawdzeniu poprawności wywołania. Proszę unikać pisania całego kodu jako element wyrażenia warunkowego.
- Program testujący powinien wyświetlać w jednym wierszu wszystkie elementy tablicy jeszcze przed jej posortowaniem. Następnie, już po posortowaniu, powinien ponownie wyświetlić zawartość posortowanej tablicy.
- Deklaracje funkcji należy umieścić w pliku nagłówkowym: **sortint.h**, zaś definicje funkcji w osobnych plikach, zgodnie z nazwami funkcji. Przykładowo, jeśli funkcja nazywa się: **bubble_int**, to plik winien nazywać się: **bubble_int.c**.
- Skompilowane funkcje należy umieścić w bibliotece której plik nazywa się: libsortint.a
- W procesie kompilacji programu testującego należy dołączać bibliotekę (używamy opcji 1 i L)

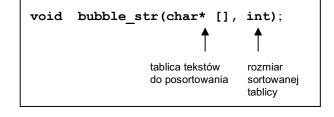
Zad. 19.

Analogicznie do zadania 18, utwórz teraz nową bibliotekę, która zawiera dwie funkcje sortujące tablice, których komórki zawierają tekst (o dowolnej długości).

Program testujący winien nosić nazwę: zad19.c

Uwaga:

- Deklaracje obu funkcji podobnie do tej w przykładzie w ramce:
- Deklaracje obu funkcji należy umieścić w pliku nagłówkowym: sortstr.h,
- Plik biblioteki powinien się nazywać: libsortstr.a
- Wykorzystaj tylko tablicę statyczną.



• Możesz utworzyć własną tablicę statyczną zawierającą z góry wprowadzone wartości lub użyć do sortowania tablicę argw (tablica argumentów wywołania), jednak z pominięciem elementów stanowiących nazwę programu i opcję jego wywołania. Jeśli zdecydujesz się na pierwszy wariant to program winien wypisać tablicę przed i po posortowaniu. W drugim przypadku wystarczy wypisać elementy po posortowaniu. Przykład wywołania dla drugiego wariantu:

```
./zad18 -n bubble Ala Ela Ola Ewa
Ala Ela Ewa Ola
./zad18 -nbubble Ala Ela Ola Ewa
Ala Ela Ewa Ola
```

Zagadnienia do przygotowania na kolejne zajęcia:

- Wskaźniki do funkcji (tworzenie i korzystanie)
- Przekazywanie wskaźnika do funkcji na przykładzie polecenia qsort (proszę zapoznać się z dokumentacją: man -s3 qsort)