ISTOTNE UWAGI DO PONIŻSZYCH ZADAŃ:

- Nie należy stosować zmiennych globalnych;
- Kod winien być pisany przejrzyście z uwzględnieniem wcięć podkreślających poziom polecenia;
- Każdy plik nagłówkowy winien zawierać stosowne dyrektywy dla kompilacji warunkowej.
- Koniecznie należy stosować dokładnie te nazwy, które podano w treści zadań.
- Wywołanie programu testującego bez parametrów spowoduje wyświetlenie składni wywołania danego programu.

Zad. 20.

Na podstawie zadań 18 i 19 utwórz nowe dwa programy testujące – odpowiednio: **`zad18a.c`**, i **`zad19a.c`** w których tablice statyczne należy zastąpić tablicami dynamicznymi. W obu przypadkach należy poprosić użytkownika o podanie rozmiaru tablicy do posortowania, a następnie należy pobrać od użytkownika wartości poszczególnych komórek tablic.

UWAGA:

Program powinien zweryfikować poprawność wprowadzonej wartości (rozmiar tablicy). W przypadku wartości niepoprawnej tj. mniejszej lub równej jeden, powinien (do skutku) oczekiwać na wprowadzenie poprawnej wartości. *(Zgodnie z posiadaną wiedzą użyj odpowiedniej pętli… Jaką pętlę stosuje się w takim przypadku?).*

Tworzenie tablic dynamicznych wymaga rezerwacji i zwalniania pamięci – zastosuj odpowiednio, znane funkcje (**`malloc`**, **`calloc`** i **`free`**). Zawsze sprawdzaj, czy udało się utworzyć tablicę i zgodnie z dobrym zwyczajem, zawsze po zwolnieniu pamięci wyzeruj wartość wskaźnika.

Zad. 21.

Po zapoznaniu się z dokumentacją do funkcji: **`qsort`** zmień implementacje dwóch wybranych wcześniej algorytmów sortujących tworząc uniwersalne funkcje sortujące. Funkcje powinny posiadać deklarację taką jak deklaracji funkcji: **`qsort`**. (Więcej w ramce.) Dla ułatwienia podano kod dwóch programów sortujących (**`zad21a.c`** i **`zad21b.c`**), które to programy sortują odpowiednio tablicę statyczną liczb całkowitych i tablicę statyczną tekstów przy zastosowaniu funkcji: **`qsort`**.
Zaimplementuj także przydatną funkcję: **`swap`**, którą wykorzystaj w implementacjach obu funkcji sortujących. Jej deklaracja: **`void swap(void *ad1, void *ad2, int bytes);`**
Funkcja tam ma zamieniać ze sobą miejscami dwa elementy o adresach: **`ad1`** i **`ad2`**, liczba bajtów za adresami: **`bytes`**. Zastanów się nad wykorzystaniem w niej funkcji: **`memcpy`**

Nowe implementacje algorytmów sortujących wraz z funkcją **`swap`** umieść w bibliotece: **`sort`** (plik biblioteki: **`libsort.a`**, plik nagłówkowy: **`sort.h`**).

Następnie utwórz kolejne programy testujące oznaczone jako: **`zad18b.c`** i **`zad19b.c`**

UWAGA:

Programy te powinny sortować odpowiednio:

- **`zad18b.c`** sortowanie dynamicznej tablicy liczb całkowitych przy użyciu funkcji z własnej biblioteki **`sort`** (plik biblioteki: **`libsort.a`**). oraz funkcji **`qsort`** (dodaj trzecią opcję sortowania). Argumenty obu funkcji sortujących powinny być takie samej jak dla funkcji: **`qsort`** *(por. z ramką poniżej)*. Program stanowi rozbudowaną postać: **`zad18a.c`**

- **`zad19b.c`** sortowanie tablicy tekstów przy użyciu funkcji z własnej biblioteki **`sort`** (plik biblioteki: **`libsort.a`**) oraz funkcji **`qsort`** (dodaj trzecią opcję sortowania). Argumenty obu funkcji sortujących powinny być takie samej jak dla funkcji: **`qsort`** *(por. z ramką poniżej)*. Program stanowi rozbudowaną postać: **`zad19a.c`**

```
Standard C Library Functions qsort(3C)

NAME
 qsort - quick sort

SYNOPSIS
 #include <stdlib.h>

 void  qsort(void * base, size_t nel, size_t width, int (* compar)(const void *, const void *));
```

zad21a.c

```
#include <stdio.h>

int comp_int(const void * ad1, const void * ad2) {
   return *(const int *) ad1 - *(const int *) ad2;
}

int main() {

   int tab[] = { 1, 3, -120, 120, 12, };

   int size = sizeof(tab) / sizeof(int);

   for (int i=0; i<size; i++)  printf("%d ", tab[i]);
   printf("\n");

   qsort(tab, size, sizeof(int), comp_int);

   for (int i=0; i<size; i++)  printf("%d ", tab[i]);
   printf("\n");

   return 0;
}
```

zad21b.c

```
#include <stdio.h>
#include <string.h>

int comp_str(const void * ad1, const void * ad2) {
   return strcmp(*(const char **) ad1, *(const char **) ad2);
}

int main() {

   char* tab[] = {"Warszawa", "Krakow", "Kielce", "Zakopane", "Bochnia"};

   int size = sizeof(tab) / sizeof(char*);

   for (int i=0; i<size; i++)  printf("%s ", tab[i]);
   printf("\n");

   qsort(tab, size, sizeof(char*), comp_str);

   for (int i=0; i<size; i++)  printf("%s ", tab[i]);
   printf("\n");

   return 0;
}
```