ЛАБОРАТОРНА РОБОТА 1
Алгоритми впорядкування (сортування)

ТЕОРЕТИЧНИЙ МАТЕРІАЛ

Зазвичай існує декілька алгоритмів розв'язання однієї і тієї самої задачі.

Складність алгоритму - функція, яка описує асимптотичну залежність кількості операцій від розміру задачі.

Задача: Пошук значення у впорядкованому цілочисловому масиві.

Пошук перебором - складність $O(n)$:

```
int directSearch(int* arr, int n, int goal){
  for(int i=1;i<=n;i++){
    if(arr[i]==goal) return i;
  }
  return -1;
}
```

Бінарний пошук - складність $O(log(n))$[1]:

```
int binarySearch(int* arr, int n, int goal){
  int low = 0;
  int high = n-1;
  while (low<=high){
    int middle = (low+high)/2;
    if (arr[middle]==goal) return middle;
    if (arr[middle]>goal) high = middle-1; else low = middle+1;
  }
  return -1;
}
```

Алгоритми впорядкування:

1. Сортування обміном (bubble sort).
2. Сортування вибором (selection sort).
3. Сортування простою вставкою (insertion sort).
4. Швидке сортування (quick sort).
5. Сортування злиттям (merge sort).
6. …

[1] Якщо основа логарифма не вказується, то вважається, що вона дорівнює 2

Ітеративна реалізація бульбашкового алгоритму (на прикладі впорядкування за зростанням цілочислового масиву):

```
void bubbleSort(int* arr, int n){
  for (int i=1; i<=n-1; i++){
    for (int j=0; j<n-i; j++){
      if (arr[j] > arr[j+1]) swap(arr[j+1], arr[j]);
    }
  }
}
```

Реалізація сортування вибором (переміщення найменшого елемента) (на прикладі впорядкування за зростанням цілочислового масиву):

```
void selectionSort(int* arr, int n){
  int i, j, minIndex;
  for (i=0; i<n-1; i++){
    minIndex = i;
    for (j=i+1; j<n; j++) {
      if (arr[j] < arr[minIndex]) {
        minIndex = j;
      }
    }
    if (minIndex != i) swap(arr[i], arr[minIndex]);
  }
}
```

Реалізація сортування простою вставкою (на прикладі впорядкування за зростанням цілочислового масиву):

```
void insertionSort(int* arr, int n){
  for (int i=1; i<n; i++) {
    int movingElement = arr[i];
    int j = i-1;
    while (j>=0 && arr[j]>movingElement) {
      arr[j+1] = arr[j];
      j--;
    }
    arr[j+1] = movingElement;
  }
}
```

Реалізація швидкого (рекурсивного) сортування (на прикладі впорядкування за зростанням цілочислового масиву):

```
void quickSort(int* arr, int start, int stop) {
  int discr = arr[start];
  int examined = start+1;
  int moved = start;
  while (examined < stop) {
    if (arr[examined] < discr) {
      int movedElement = arr[examined];
      for (int i=examined; i>moved; --i) arr[i] = arr[i-1];
      arr[moved] = movedElement;
      ++moved;
    }
    ++examined;
  }
  if (moved > 0 && start < moved-1) quickSort(arr, start, moved);
  if (moved+1 < stop) quickSort(arr, moved+1, stop);
}

void quickSort(int* arr, int n) {quickSort(arr, 0, n);}
```

Реалізація сортування злиттям (на прикладі впорядкування за зростанням цілочислового масиву):

```
void copyArray(int* source, int* dest, int n) {
     for (int i=0; i<n; i++) dest[i] = source[i];
}

void merge(int* a, int size_a, int* b, int size_b, int* c) {
  int i=0, j=0, k=0;
  while (true) {
    if (a[i] < b[j]) {
      c[k++] = a[i++];
      if (i == size_a) {
        while (j < size_b) c[k++] = b[j++];
        break;
      }
    }
    else {
      c[k++] = b[j++];
      if (j == size_b) {
        while (i < size_a) c[k++] = a[i++];
        break;
      }
```

```
    }
  }
}

void mergeSort(int* arr, int n) {
  int* newArray = new int[n];
  bool mergeToNew = true;
  int len = 1;
  while (len < n) {
    int k = 0;
    int times = n / (2*len);
    if (mergeToNew) {
      for (int i=0; i<times; ++i) {
        merge(arr+k, len, arr+k+len, len, newArray+k);
        k += 2*len;
      }
      if (k < n) {
        if (k+len < n)
          merge(arr+k, len, arr+k+len, n-k-len, newArray+k);
        else
          copyArray(arr+k, newArray+k, n-k);
      }
    }
    else {
      for (int i=0; i<times; ++i) {
        merge(newArray+k, len, newArray+k+len, len, arr+k);
        k += 2*len;
      }
      if (k < n) {
        if (k+len < n)
          merge(newArray+k, len, newArray+k+len, n-k-len, arr+k);
        else
          copyArray(newArray+k, arr+k, n-k);
      }
    }
    len *= 2;
    mergeToNew = !mergeToNew;
  }
  if (!mergeToNew) copyArray(newArray, arr, n);
  delete[] newArray;
}
```

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Реалізувати у вигляді функції алгоритм сортування згідно з Вашим номером варіанту та продемонструвати його на прикладі масивів. Передбачити вибір напрямку впорядкування (за зростанням чи спаданням) та виведення проміжних результатів. Оголошення функцій розташовувати у відповідних заголовкових та cpp-файлах.

ВАРІАНТИ ЗАВДАНЬ ДЛЯ ІНДИВІДУАЛЬНОГО ВИКОНАННЯ:

1. Сортування змішуванням (cocktail sort).
2. Сортування гнома (gnome sort).
3. Сортування гребінцем (comb sort).
4. Сортування Шелла (Shellsort).
5. Сортування вибором (insertion sort) з переміщенням найбільшого елемента.
6. Блукаюче сортування (stooge sort).
7. Сортування підрахунком (counting sort).
8. Впорядкування за розрядами (LSD Radix sort).
9. Бітонічне сортування (bitonic mergesort).
10. Сортування простими вставками з бінарним пошуком.
11. Ниткоподібне сортування (strand sort).
12. Сортування вставками зі сторожовим елементом (бар’єром).
13. Парне сортування простими вставками.
14. Сортування Діріхле.
15. Гравітаційне сортування (bead sort).
16. Сортування перестановками (permutation sort).
17. Млинцеве сортування (pancake sort).
18. Сортування комірками (bucket sort).
19. Рекурсивне сортування обміном (recursive bubble sort).