

POLITECHNIKA ŚLĄSKA W GLIWICACH

WYDZIAŁ ELEKTRYCZNY

Katedra Energoelektroniki, Napędu Elektrycznego i Robotyki

**PROJEKT INŻYNIERSKI**

**Równoległe algorytmy sortowania**

**Parallel sorting algorithms**

Student: **Marek Jacek TOMCZYK**  
Nr albumu: 265698

Studia: Stacjonarne I stopnia  
Kierunek: Informatyka

Specjalność: Informatyka użytkowa

Promotor: dr inż. Marcin POŁOMSKI

# SPIS TREŚCI

[SPIS TREŚCI 1](#_Toc345587350)

[Rozdział 1 - Wstęp teoretyczny 2](#_Toc345587351)

[1.1 – Cel projektu 2](#_Toc345587352)

[1.2 – Założenia 3](#_Toc345587353)

[Rozdział 2 – Narzędzia programowe użyte przy tworzeniu systemu 4](#_Toc345587354)

[2.1 – Technologie i narzędzia 4](#_Toc345587355)

[2.1.1 – Języki programowania 4](#_Toc345587356)

[2.1.2 – Przechowywanie danych 5](#_Toc345587357)

[2.1.3 – Dodatkowe programy i narzędzia 5](#_Toc345587358)

# Rozdział 1 - Wstęp teoretyczny

## 1.1 – Cel projektu

Celem projektu jest implementacja oraz analiza porównawcza wybranych algorytmów sortowania, a także praktyczne zastosowanie języka C++ oraz wykorzystanie mechanizmów jakie oferuje standard języka C++17. Realizacja projektu ma charakter badawczy – porównanie czasów potrzebnych na sortowanie obszernych zbiorów danych. W następnym rozdziale przedstawione zostaną podstawowe informacje dotyczące tematyki projektu, takie jak proces, wątek czy system operacyjny, a także opis wybranych algorytmów wraz z przykładową implementacją w języku C++. Następne rozdziały przedstawiają architekturę obiektową aplikacji, zastosowane mechanizmy ze standardu C++17 wraz z przykładami użycia. W kolejnych rozdziałach zostaną przedstawione wyniki badań w postaci wykresów oraz wnioski.

## 1.2 – Wprowadzenie teoretyczne

*System operacyjny (ang. Operating System [OS])[1]* – program zarządzający sprzętem komputerowym. Można go traktować jako łącznik pomiędzy programem użytkownika a urządzeniami wejścia/wyjścia. System jest swoistym programem sterującym, który nadzoruje wykonywanie programu użytkownika, przeciwdziała błędom oraz obsługuje urządzenia.

*Proces (ang. Process)[1]* – egzemplarz wykonywanego programu. Aplikacja może składać się z większej liczby procesów. Każdy proces otrzymuje unikatowy numer, zwany PID (od ang. Process Identifier), który jednoznacznie go identyfikuje. System operacyjny przydziela każdemu procesowi jasno określone zasoby. Program może zażądać utworzenia określonej liczby wątków, wykonujących wskazane części programu. Wątki współdzielą wszystkie zasoby zarezerwowane dla procesu z wyjątkiem czasu procesora, który jest indywidualnie przydzielany każdemu wątkowi.

*Wątek (ang. Thread)[1]*– część programu wykonywana współbieżnie w obrębie jednego procesu – w jednym procesie może istnieć wiele wątków. Różnica między zwykłym procesem, a wątkiem polega na współdzieleniu przez wszystkie wątki działające w danym procesie przestrzeni adresowej oraz wszystkich innych struktur systemowych (np. listy otwartych plików, gniazd itp.) – z kolei procesy posiadają niezależne zasoby. Dzięki temu wątki wymagają mniej zasobów do działania i też jest mniejszy czas ich tworzenia.

*Wielowątkowość (ang. Multithreading)[1]* – jest to cecha systemu operacyjnego, dzięki której w ramach jednego procesu może być wykonywanych kilka zadań jednocześnie – nazywanych wątkami. Każde zadanie określa się jako ciągi instrukcji realizowane do pewnego stopnia niezależnie. Wszystkie wątki w ramach tego samego procesu współdzielą tą samą wirtualną przestrzeń adresową – kod programu oraz jego dane.

## 1.3 - Wybrane algorytmy sortowania

Sortowanie jest czynnością wykonywaną codziennie i praktycznie w każdej dziedzinie życia – czy to w szeroko pojętej informatyce, jak również w mechanice, elektronice. Algorytmów sortujących jest bardzo wiele jednak do badań zostały wybrane trzy algorytmy – *Bitonic sort*, *Quick sort* oraz *Merge sort*. Poniżej znajduje się opis każdego z wcześniej wymienionych algorytmów wraz z przykładową implementacją w języku C++.

### 1.3.1 – Bitonic sort

Bitonic sort to równoległy algorytm sortowania, który jest również wykorzystywany do tworzenia sieci sortujących. Algorytm został opracowany przez Ken’a Batcher. Powstałe sieci sortujące składają się z komparatorów i mają opóźnienie , gdzie jest liczbą elementów do posortowania.

**Listing 1. Przykładowa implementacja algorytmu Bitonic sort w języku C++**

**void** bitonicMerge(**int** arr[], **int** low, **int** count, **bool** asc)

{

**if**( count > 1 )

{

**int** k = count / 2;

**for**( **int** i = low; i < low + k; i++)

{

**if**( asc == (arr[i] > arr[i + k]) )

{

**int** tmp = arr[i];

arr[i] = arr[i + k];

arr[i + k] = tmp;

}

}

bitonicMerge(arr, low, k, asc);

bitonicMerge(arr, low + k, k, asc);

}

}

**void** bitonicSort(**int** arr[], **int** low, **int** count, **bool** asc)

{

**if**( count > 1 )

{

**int** k = count / 2;

bitonicSort(arr, low, k, **true**);

bitonicSort(arr, low + k, k, **false**);

bitonicMerge(arr, low, count, asc);

}

}

### 

### 1.3.2 – Quick sort

Quicksort jest jednym z najpopularniejszych algorytmów sortowania działających na zasadzie *„dziel i zwyciężaj”.* Algorytm został wynaleziony przez C.A.R. Hoare’a. Średnia złożoność obliczeniowa tego algorytmu jest rzędu , a ze względu na szybkość i prostotę implementacji jest powszechnie używany. Jego implementacje znajdują się w bibliotekach standardowych wielu środowisk programowania.

**void** quickSort(**int** arr[], **int** low, **int** high)

{

**if**( low < high )

{

**int** pivot = arr[high];

**int** i = (low - 1);

**for**(**int** j = low; j <= (high - 1); j++)

{

**if**( arr[j] <= pivot)

{

i++;

**int** temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

**int** temp = arr[i + 1];

arr[i + 1] = arr[high];

arr[high] = temp;

**int** partIndex = (i + 1);

quickSort(arr, low, partIndex - 1);

quickSort(arr, partIndex + 1, high);

}

}

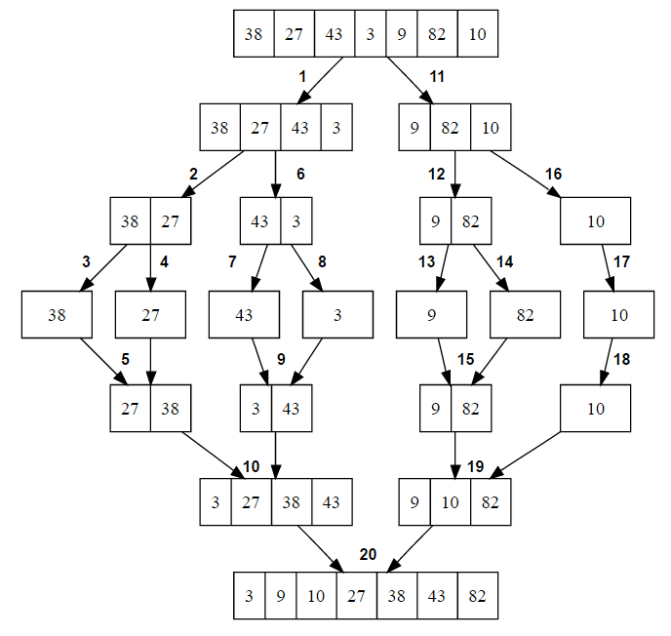
**Listing 2. Przykładowa implementacja algorytmu Quicksort w języku C++**

### 1.3.3 – Merge sort

Sortowanie przez scalanie (ang. Merge sort) – rekurencyjny algorytm sortowania danych, stosujący metodę „dziel i zwyciężaj”. Odkrycie algorytmu przypisuje się Johnowi von Neumannowi. Wyróżnić można trzy podstawowe kroki działania algorytmu:

1.Podziel zestaw danych na dwie równe części  
2.Zastosuj sortowanie przez scalanie dla każdej z nich oddzielnie   
3.Połącz posortowane podciągi w jeden ciąg posortowany

Podstawową wersję algorytmu sortowania przez scalanie można uprościć poprzez odwrócenie procesu scalania serii. Ciąg danych możemy wstępnie podzielić na serii długości , scalić je tak by otrzymać serii długości 2, scalić je otrzymując serii długości 4 itd. Złożoność obliczeniowa jest taka sama jak w przypadku klasycznym jednak, dzięki zrezygnowaniu z rekursji oszczędzamy czas oraz pamięć potrzebną na jej obsłużenie.



**Rys. 1. Schemat działania algorytmu Mergesort** (źródło: techiedelight.com/merge-sort)

# *Rozdział 2 – Narzędzia programowe użyte przy tworzeniu systemu*

W tym rozdziale przedstawione zostaną narzędzia, użyte podczas tworzenia projektu. Każde z narzędzi zostanie krótko opisane wraz z uzasadnieniem dlaczego zostało wybrane.

## 2.1 – Technologie i narzędzia

### 2.1.1 – Języki programowania

Cały projekt został napisany w języku C++ w standardzie C++17. Został wybrana z powodu:

- Dużego zainteresowania wśród programistów,

- Szybkości działania,

- Ciągłego rozwoju i rozbudowywania funkcjonalności języka,

- Braku opłat licencyjnych,

- Chęci poznania standardu C++17 i jego możliwości,

- Powszechnej dostępności materiałów.

Język C++ jest ciągle jednym z najbardziej popularnych języków programowania na świecie, wg. portalu Tiobe znajduje się on na 4 pozycji, zaraz za Javą, językiem C oraz Python. Język C++ swoje największe zastosowanie znajduje w systemach wbudowanych oraz tworzeniu gier komputerowych, czyli w dziedzinach gdzie liczy się łatwy dostęp do sprzętu, a także szybkość.

### 2.1.2 – Przechowywanie danych

Aplikacja zapisuje posortowane porcje danych w osobnych plikach na dysku użytkownika. Są to dane tymczasowe, potrzebne na posortowanie dużych zestawów danych. Pliki zostaną usunięte po zakończeniu działania aplikacji.

### 

### 2.1.3 – Dodatkowe programy i narzędzia

* **Git** – rozproszony system kontroli wersji, którego twórcą jest Linus Torvalds – twórca jądra systemu Linux. Dzięki tej technologii łatwiej zarządzać projektem, przeglądać postępy, wprowadzone zmiany, a także dzielić problemy na mniejsze - rozwiązywane na osobnych gałęziach, tym samym zachowując stabilną wersję programu.
* **Trello** – system online wspomagający zarządzanie projektem, dzięki któremu łatwiej rozplanować prace, rozdzielić je na kilka etapów oraz