PENGEMBANGAN ALAT BANTU PEMBELAJARAN *SORTING ALGORITHM* BERBASIS VISUAL *CONSOLE* C++

Puspa Dwi Setyorini[1], Lintang Tsaniatu Azzahro[2], Ramona Aprilia Yuniar[3], Imam Prayogo Pujiono[4]
Program Studi Informatika, Fakultas Ekonomi dan Bisnis Islam
Universitas Islam Negeri K.H. Abdurrahmah Wahid
Jalan Kusuma Bangsa No 09, Panjang Baru, Kec. Pekalongan Utara, Kota Pekalongan
puspa.dwi.setyorini24003@mhs.uingusdur.ac.id[1],
lintang.tsaniatu.azzahro24007@mhs.uingusdur.ac.id[2],
ramona.aprilia.yuniar24039@mhs.uingusdur.ac.id[3]
imam.prayogopujiono@uingusdur.ac.id[4]

Abstrak

Pemahaman terhadap algoritma pengurutan seperti *Bubble sort* dan *Selection Sort* kerap menjadi tantangan bagi pemula, khususnya mahasiswa baru yang berada dalam tahap awal pembelajaran struktur data. Visualisasi terbukti menjadi metode yang efektif untuk membantu pemula memahami proses algoritma secara lebih intuitif. Penelitian ini bertujuan mengembangkan alat bantu pembelajaran berbasis visual console menggunakan bahasa C++ tanpa ketergantungan pada pustaka grafis eksternal. Visualisasi ditampilkan melalui simbol teks (karakter # dan *) di *console*, yang merepresentasikan elemen *array* dan proses perbandingan antar elemen. Alat ini dirancang untuk menunjukkan secara interaktif langkah-langkah eksekusi algoritma pengurutan dengan penambahan jeda waktu (*delay*) antar proses guna mempermudah pemahaman. Hasil pengujian terhadap 40 mahasiswa menunjukkan bahwa mayoritas merasa terbantu dalam memahami konsep pengurutan melalui pendekatan ini. Pendekatan visual berbasis teks dianggap cukup efektif, sederhana, dan mudah diakses dalam lingkungan pembelajaran dasar. Secara keseluruhan, visualisasi *console* ini terbukti mampu memperkuat pemahaman konseptual mahasiswa terhadap algoritma pengurutan meskipun tanpa menggunakan antarmuka grafis yang kompleks.

Kata Kunci : Visualisasi Algoritma, Pengurutan, C++, Console, Pembelajaran Interaktif

Abstract

*Understanding sorting algorithms such as Bubble sort and Selection Sort is often a challenge for beginners, especially freshmen who are in the early stages of learning data structures. Visualization is proven to be an effective method to help beginners understand the algorithm process more intuitively. This research aims to develop a visual console-based learning tool using the C++ language without dependence on external graphics libraries. Visualizations are displayed through text symbols (# and * characters) in the console, which represent array elements and the comparison process between elements. The tool is designed to interactively show the execution steps of the sorting algorithm with the addition of delays between processes to facilitate understanding. The results of testing 40 students showed that the majority found it helpful in understanding the concept of sorting through this approach. The text-based visual approach is considered effective, simple, and accessible in a basic learning environment. Overall, this console visualization proved to be able to strengthen students conceptual understanding of the sorting algorithm even without using a complex graphical interface.*

***Keyword** : Algorithm Visualization, Sorting, C++, Console, Interactive Learning*

PENDAHULUAN

Algoritma pengurutan (*sorting algorithm*) merupakan konsep dasar yang penting dalam ilmu komputer karena memiliki banyak aplikasi praktis, seperti penyusunan data dalam basis data, pencarian data, dan optimisasi. Meskipun algoritma seperti *Bubble sort* dan *Selection Sort* telah umum diajarkan, tantangan

utama dalam pembelajarannya adalah bagaimana menyampaikan proses kerja algoritma tersebut secara efektif, khususnya kepada pemula. Pemahaman mendalam terhadap algoritma pengurutan juga akan sangat membantu mahasiswa dalam mempelajari struktur data dan algoritma yang lebih kompleks (Cormen et al., 2022).
dan *Selection Sort* menggunakan bahasa C++, yang diharapkan mampu meningkatkan pemahaman mahasiswa terhadap proses pengurutan tanpa ketergantungan pada pustaka grafis eksternal.

METODE PENELITIAN

Penelitian ini merupakan jenis penelitian pengembangan (research and development) yang bertujuan untuk merancang dan membangun alat bantu visualisasi algoritma pengurutan berbasis teks pada *console* menggunakan bahasa pemrograman C++. Alat bantu ini dikembangkan untuk membantu mahasiswa memahami cara kerja algoritma *Bubble sort* dan *Selection Sort* secara visual dan interaktif.
Penelitian ini mengacu pada pendekatan literatur dan desain yang umun digunakan dalam pengembangan aplikasi edukatif berbasis terminal (N. T. S. Saptadi et al., 2024)

1. Tahapan Pengembangan
   Metode pengembangan dilakukan melalui tahapan berikut:
   Analisis Kebutuhan: Mengidentifikasi kendala mahasiswa dalam memahami proses pengurutan secara manual dan keterbatasan dalam menggunakan pustaka visualisasi grafis (Halimatussya'diyah Purba & Yahfizham Yahfizham, 2023). Perancangan: Mendesain sistem visualisasi yang ditampilkan di console menggunakan karakter # dan * untuk menggambarkan nilai dan posisi elemen yang sedang dibandingkan.
   Implementasi: Menggunakan bahasa C++ tanpa pustaka eksternal. Dataset berisi 10 elemen angka acak dalam array (A. H. Saptadi & Sari, 2012). Pengujian: Pengujian dilakukan secara internal dengan mengamati tampilan output visual secara berulang untuk memastikan proses pengurutan berjalan sesuai logika algoritma.

2. Potongan Kode Utama
   Salah satu fungsi utama adalah printArray, yang menampilkan array dalam bentuk bar visual menggunakan karakter:

```
void printArray(const vector<int>& arr, int highlight1 = -1, int highlight2 = -1) {
   for (int i = 0; i < arr.size(); i++) {
      for (int j = 0; j < arr[i]; j++) {
         if (i == highlight1 || i == highlight2)
            cout << "*";
         else
            cout << "#";
      }
      cout << " (" << arr[i] << ")" << endl;
   }
   cout << "--------------------" << endl;
}
```

   Fungsi delay digunakan untuk memberi jeda waktu antar proses agar perubahan posisi elemen dapat diamati dengan jelas:

```
void delay(int ms) {
   this_thread::sleep_for(chrono::milliseconds(ms));
}
```

   Pendekatan visual ini tidak menggunakan pustaka grafis tambahan seperti SFML atau SDL, sehingga lebih ringan dan sesuai untuk lingkungan pembelajaran dasar. Representasi dengan

simbol teks memungkinkan pengguna mengamati proses pengurutan secara langsung dan intuitif.

3. Alasan Pemilihan Algoritma
   Algoritma *Bubble sort* dan *Selection Sort* dipilih karena merupakan algoritma dasar yang umum diajarkan pada tahap awal pembelajaran struktur data. Keduanya memiliki logika iteratif yang mudah divisualisasikan dan dipahami oleh pemula.

HASIL DAN PEMBAHASAN

Penelitian ini menghasilkan sebuah alat bantu pembelajaran berupa program computer yang mampu memvisualisasikan proses kerja dari dua algoritma pengurutan data yang sangat umum digunakan dalam dunia pemprograman, yaiti algoritma *Bubble sort* dan *Selection Sort.* program ini dikembangkan menggunakan Bahasa pemprograman C++ dan dirancang agar dapat dijalankan pada lingkungan console, tanpa bergantung pada pustaka grafis tambahan. Pendekatan visualisasi yang digunakan cukup sederhana, yakni dengan menampilkan elemen array dalam bentuk karakter ASCII, dimana symbol # dan * berfungsi sebagai representasi dari nilai dan posisi elemen yang sedang dibandingkan. Dengan car ini, proses pengurutan yang bersifat abstrak menjadi lebih konkret dan mudah dipahami oleh mahasiswa atau pengguna pemula (Goswami, 2025).

1. Implementasi *Bubble sort*
   Cuplikan kode:

```
void bubbleSort(vector<int>& arr) {
   for (int i = 0; i < arr.size() - 1; i++) {
      for (int j = 0; j < arr.size() - i - 1; j++) {
         printArray(arr, j, j + 1);
         delay(300);
         if (arr[j] > arr[j + 1]) {
            swap(arr[j], arr[j + 1]);
         }
      }
   }
   printArray(arr);
}
```

   Penjelasan:
   Algoritma *Bubble sort* bekerja dengan cara melakukan perbandingan antar pasangan elemen yang berdekatan dalam array, kemudian menukarnya jika elemen di posisi kiri lebih besar dari elemen array di posisi kanan. Proses ini diulang treus-menerus hingga seluruh elemen berada dalam urutan yang benar. Dalam visualisasi yang dikembangkan, dua elemen yang sedang dibandingkan ditandai dengan dimbol *, sedangkan elemen lain yang tidak sedang dibandingkan tetap ditampilkan sebagai #. Hal ini membantu pengguna untuk melacak proses tukar-menukar elemen secara visual. Pendekatan visualisasi berbasis teks ini telah digunakan dalam berbagai aplikasi edukatif, seperti pada proyek open source yang tersedia di GitHub, yang menunjukkan efektivitas pendekatan ini dalam membantu pemahaman konsep algoritma pengurutan (GeeksforGeeks, 2023).

2. Implementasi *Selection Sort*
   Cuplikan kode:

```
void selectionSort(vector<int>& arr) {
   for (int i = 0; i < arr.size() - 1; i++) {
      int minIndex = i;
      for (int j = i + 1; j < arr.size(); j++) {
         printArray(arr, minIndex, j);
         delay(300);
         if (arr[j] < arr[minIndex]) {
            minIndex = j;
         }
      }
      swap(arr[i], arr[minIndex]);
   }
   printArray(arr);
}
```

Penjelasan:
Berbeda dengan *Bubble sort*, algoritma *Selection Sort* bekerja dengan cara mencari elemen terkecil dari bagian array yang belum diurutkan, lalu menempatkannya ke posisi yang sesuai di awal array. Proses ini dilakukan berulang hingga seluruh elemen berada dalam posisi yang benar. Dalam program in, elemen yang sedang dibandingkan dan elemen minimun yang sedang dicari ditamilkan dengan tanda *, yang memberi petunjuk visual kepada pengguna bahwa proses seleksi dan pertukaran sedang berlangsung.

3. Contoh Data Input
   vector<int> data = {5, 1, 4, 2, 8, 7, 3, 10, 6, 9};
   Data ini digunakan untuk menguji jalannya visualisasi dari kedua algoritma pengurutan, dan hasil pengurutan ditampilan secara bertahap agar prosesnya dapat diamati secara langsung.

4. Tabel Perbandingan Karakteristik

Tabel 1. Perbandingan Karakteristik

No	Kriteria	*Bubble Sort*	*Selection Sort*
1	Kompleksitas Waktu	$O(n^2)$	$O(n^2)$
2	Jumlah Pertukaran	Relatif lebih banyak	Lebih sedikit karena hanya satu per iterasi
3	Perbandingan Elemen	Dilakukan berkali-kali per iterasi	Dilakukan hingga menemukan elemen minimum
4	Tingkat Interaktivitas	Sangat tinggi dan animatif	Sedang, namun lebih sistematis
5	Kesesuaian untuk Pemula	Sangat baik untuk memperlihatkan dasar pengurutan	Baik untuk memperlihatkan proses seleksi

Notasi **$O(n^2)$** (dibaca: "Big O dari n kuadrat") merupakan cara untuk menggambarkan kompleksitas waktu dari suatu algoritma, khususnya bagaimana **waktu eksekusi algoritma meningkat** seiring bertambahnya jumlah data input, yang dilambangkan dengan **n**. Dalam

konteks **O(n²)**, ini berarti bahwa jumlah langkah atau operasi yang dilakukan algoritma **bertambah secara kuadrat** terhadap ukuran inputnya.

Jika suatu algoritma memiliki kompleksitas **O(n²)**, artinya:

a) Ketika jumlah data **n** bertambah dua kali lipat, maka waktu eksekusinya akan bertambah **sekitar empat kali lipat** (karena $2^2 = 4$).
b) Jika n = 10, maka kira-kira ada 100 operasi yang terjadi.
c) Jika n = 1000, maka bisa ada sekitar 1.000.000 operasi.

Cuplikan kode :

```
for (int i = 0; i < n; i++) {        // loop pertama (n kali)
   for (int j = 0; j < n; j++) {     // loop kedua (n kali untuk setiap i)
      // operasi perbandingan
```

Penjelasan :

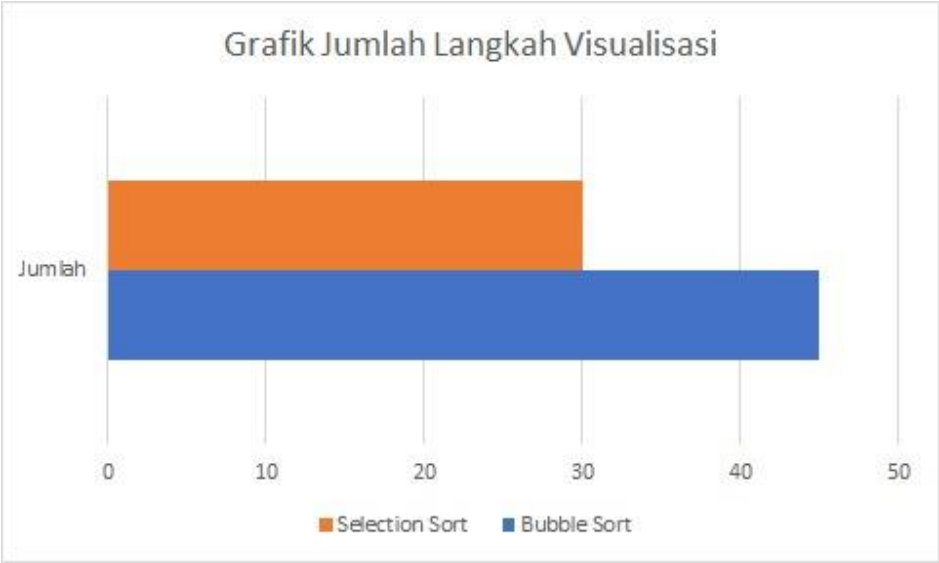
Pada potongan kode tersebut, `for (int i = 0; i < n; i++)` merupakan loop pertama yang akan berjalan sebanyak *n* kali, di mana *n* adalah jumlah elemen dalam array. Loop ini bertanggung jawab untuk mengulangi proses sebanyak jumlah elemen yang ada. Kemudian, di dalam loop pertama terdapat loop kedua, yaitu `for (int j = 0; j < n; j++)`, yang juga akan berjalan sebanyak *n* kali untuk setiap satu kali iterasi dari loop pertama. Hal ini berarti bahwa total iterasi yang terjadi secara keseluruhan adalah sebanyak $n \times n$ atau n^2 kali. Di dalam kedua loop ini biasanya ditempatkan logika yang berkaitan dengan proses pengurutan, seperti membandingkan nilai elemen array dan melakukan pertukaran posisi jika diperlukan. Struktur seperti ini umum ditemukan dalam algoritma pengurutan sederhana seperti Bubble Sort dan Selection Sort, yang memang mengandalkan perbandingan berulang antar elemen untuk menyusun data secara terurut.

$O(n^2)$ menunjukkan bahwa algoritma **tidak efisien untuk data besar.** Saat ukuran data tumbuh, waktu yang dibutuhkan meningkat sangat cepat. Misalnya, untuk array berukuran 1000, algoritma $O(n^2)$ bisa memerlukan hingga 1 juta langkah. Karena itu, algoritma dengan kompleksitas $O(n^2)$ hanya cocok untuk kasus-kasus sederhana atau data kecil (misalnya $n \leq 1000$).

Algoritma seperti Bubble Sort dan Selection Sort termasuk dalam kategori kompleksitas waktu $O(n^2)$ karena cara kerjanya yang melibatkan dua buah loop bersarang (nested loops). Loop pertama berfungsi untuk menelusuri setiap elemen dalam array secara berurutan dari awal hingga akhir. Sementara itu, untuk setiap iterasi dari loop pertama, terdapat loop kedua yang melakukan perbandingan terhadap elemen-elemen lain dalam array, baik untuk menemukan elemen terbesar atau terkecil (dalam Selection Sort), maupun untuk menukar elemen yang tidak berada dalam urutan yang benar (dalam Bubble Sort). Kombinasi dari dua loop ini menyebabkan jumlah operasi yang dilakukan algoritma meningkat secara kuadrat terhadap jumlah elemen yang diproses, sehingga menghasilkan kompleksitas waktu $O(n^2)$.

 e-ISSN : 2715-8756

5. Grafik Jumlah Langkah Visualisasi (Simulasi)



Grafikl 1. Jumlah Langkah Visualisasi

Jumlah Langkah diukur dari beberapa kali fungsi printArray() dipanggil selama eksekusi, yang merepresentasikan setiap kali visual ditampilkan di layar.

6. Evaluasi Pengguna Berdasarkan Kuesioner
Untuk mengukur efektivitas dan kemudahan penggunaan alat bantu pembelajaran visualisasi algoritma ini, dilakukan uji coba terhadap 40 mahasiswa baru yang diminta mencoba langsung aplikasi visualisasi algoritma pengurutan berbasis *console*, kemudian mengisi kuisioner untuk memberikan umpan balik. Berikut ringkasan hasil tanggapan mereka:

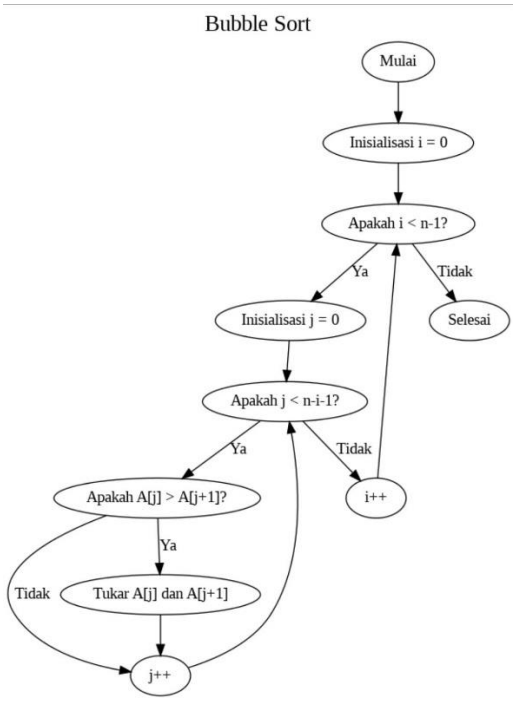
Tabel 2. Ringkasan Hasil Tanggapan Responden

No	Aspek Penilaian	Jawaban Dominan	Presentase (Estimasi)
1	Alat bantu membantu memahami pengurutan	Cukup terbantu / Sangat Terbantu	±88% menyatakan terbantu
2	Visualisasi berbasis teks mudah dipahami	Cukup Mudah	±80%
3	Symbol # dan * cukup representatif	Setuju	±76%
4	Penambahan delay antar Langkah membantu pemahaman	Lumayan Membantu	±84%
5	Algoritma yang paling mudah dipahami dan visualisasi	Keduanya (Bubble & Selection)	±56%
6	Cocok digunakan dalam perkuliahan pemprograman dasar	Cukup Cocok/Sangat Cocok	±88%

Interpretasi Temuan:

a. Mayoritas responden merasa alat bantu ini efektif dan mudah digunakan, bahkan dalam bentuk teks console.
b. *Bubble sort* dan *Selection Sort* dinilai sama-sama mudah dipahami melalui pendekatan visual yang digunakan.
c. Simbol ASCII dan Animasi baris (#, *) berhasil mewakili proses algoritma secara cukup jelas, meskipus tampilannya sangat sederhana.
d. Tambahan delay antar Langkah proses ternyata berperan penting dalam memperjelas alur visualisasi.

7. *Flowchart* Proses
   a. *Bubble sort*



Gambar 1. *Flowchart Bubble sort*

Gambar 1 menunjukkan diagram alir (*flowchart*) dari algoritma *Bubble sort*. Algoritma ini merupakan salah satu metode pengurutan yang paling sederhana, di mana proses pengurutan dilakukan dengan cara membandingkan elemen-elemen yang berdekatan dan menukarnya jika berada dalam urutan yang salah.

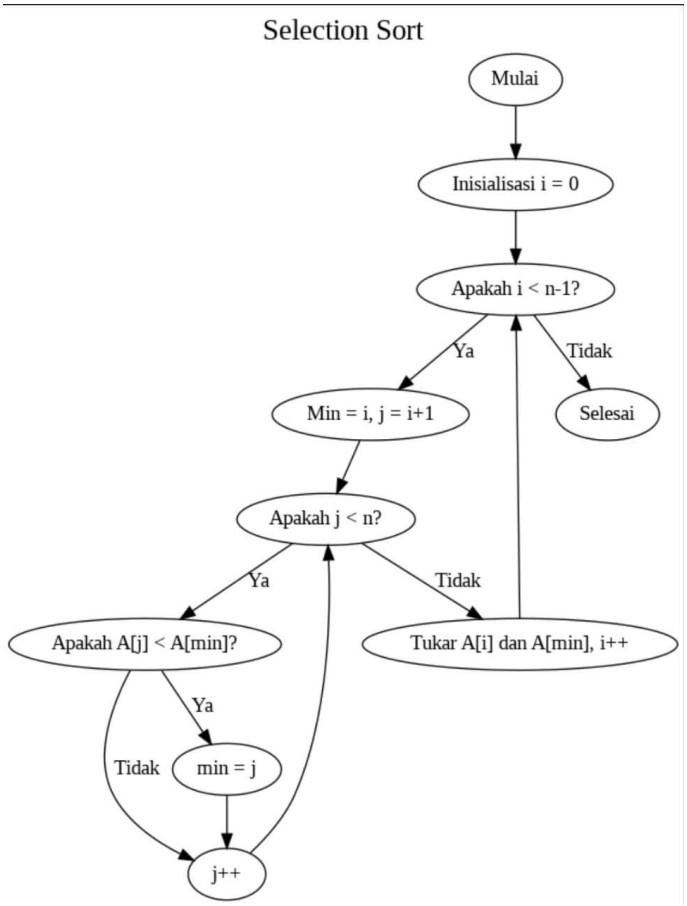
Langkah-langkah proses pada *flowchart* adalah sebagai berikut:

1) Mulai: Proses pengurutan dimulai.
2) Inisialisasi variabel i = 0: Variabel i digunakan untuk melacak jumlah iterasi yang telah dilakukan.
3) Kondisi i < n-1?: Mengecek apakah proses iterasi perlu dilanjutkan. Jika tidak, maka proses selesai.
4) Inisialisasi j = 0: Variabel j digunakan untuk membandingkan elemen pada array.

5) Kondisi j < n-i-1?: Memastikan pembandingan dilakukan pada elemen yang belum terurut.
6) Perbandingan A[j] > A[j+1]?: Jika elemen saat ini lebih besar dari elemen berikutnya, maka dilakukan pertukaran.
7) Tukar A[j] dan A[j+1]: Elemen-elemen ditukar untuk memperbaiki urutan.
8) j++: Perpindahan ke elemen berikutnya.
9) i++: Setelah satu putaran selesai, iterasi dilanjutkan.

Flowchart ini mengilustrasikan bahwa *Bubble sort* akan terus melakukan perulangan hingga seluruh elemen dalam array berada dalam urutan yang benar. Kelebihan dari algoritma ini adalah implementasinya yang sederhana, namun memiliki kekurangan dalam efisiensi waktu karena kompleksitasnya yang mencapai $O(n^2)$.

b. *Selection Sort*



Gambar 2. *Flowchart Selection Sort*

Pada Gambar 2 ditampilkan *flowchart* algoritma *Selection Sort*. Algoritma ini melakukan pengurutan dengan mencari elemen terkecil dari elemen yang belum terurut dan menempatkannya di posisi yang sesuai secara bertahap.
Tahapan dalam *flowchart* ini meliputi:

1) Mulai: Proses pengurutan dimulai.
2) Inisialisasi i = 0: Digunakan untuk menentukan posisi elemen yang akan diisi dengan elemen terkecil.

3) Kondisi i < n-1?: Mengecek apakah pengurutan masih perlu dilakukan.
4) Inisialisasi min = i, j = i+1: Menentukan indeks minimum sementara dan indeks perbandingan.
5) Kondisi j < n?: Menentukan apakah masih ada elemen yang harus dibandingkan.
6) Perbandingan A[j] < A[min]?: Jika elemen saat ini lebih kecil dari elemen minimum sebelumnya, maka:
7) min = j: Posisi elemen minimum diperbarui.
8) j++: Lanjutkan ke elemen berikutnya.
9) Setelah pencarian elemen minimum:
10) Tukar A[i] dan A[min]: Menempatkan elemen terkecil ke posisi ke-i.
11) i++: Lanjut ke iterasi berikutnya.

Selection Sort memiliki keunggulan dalam jumlah pertukaran yang lebih sedikit dibandingkan *Bubble sort*, karena pertukaran hanya dilakukan satu kali per iterasi. Namun, kompleksitas waktu tetap $O(n^2)$, menjadikannya kurang efisien untuk data berukuran besar.

8. Interpretasi Hasil

Dari hasil pengujian dan visualisasi program yang telah dikembangkan, dapat disimpulkan bahwa penggunaan simbol teks sederhana seperti # untuk merepresentasikan elemen array dan * untuk menunjukkan elemen yang sedang dibandingkan mampu memberikan efek visual yang cukup kuat dalam membangun pemahaman konseptual mahasiswa terhadap mekanisme kerja algoritma pengurutan (Mukasheva et al., 2023). Meskipun tidak menggunakan antarmuka grafis modern seperti GUI atau animasi berbasis library eksternal, visualisasi berbasis console ini tetap memberikan pengalaman belajar yang menarik dan edukatif, khususnya bagi pemula yang masih beradaptasi dengan logika algoritmik dasar.

Dalam hal interaktivitas, algoritma Bubble sort menawarkan jumlah langkah visualisasi yang lebih banyak karena prosesnya yang melibatkan perbandingan dan pertukaran elemen berulang kali. Hal ini membuatnya lebih dinamis saat divisualisasikan, sehingga sangat cocok digunakan sebagai alat bantu pembelajaran awal. Sebaliknya, algoritma Selection Sort cenderung lebih sistematis dengan jumlah pertukaran yang lebih sedikit, namun tetap memberikan ilustrasi yang jelas tentang konsep pencarian nilai minimum dan pengurutan progresif.

Dengan demikian, kedua algoritma memiliki keunggulan masing-masing yang saling melengkapi. Bubble sort ideal untuk memperkuat pemahaman tentang perbandingan dan pertukaran data, sedangkan Selection Sort lebih menekankan pada pemahaman alur logis pemilihan elemen. Keberhasilan visualisasi ini menunjukkan bahwa pendekatan pembelajaran berbasis console tidak kalah efektif dibanding visualisasi berbasis grafis, terutama dalam konteks pendidikan dasar pemrograman yang menekankan logika algoritma dan pemahaman bertahap.

SIMPULAN

Penelitian ini berhasil merancang dan mengimplementasikan alat bantu pembelajaran algoritma pengurutan berbasis visual console menggunakan bahasa pemrograman C++. Visualisasi dilakukan melalui representasi simbol teks berupa karakter # dan * yang menampilkan proses perbandingan dan pertukaran elemen dalam array secara interaktif. Tanpa ketergantungan pada pustaka grafis eksternal, pendekatan ini terbukti efektif dan ringan dijalankan di berbagai lingkungan pembelajaran.

Penambahan jeda waktu (delay) antar proses juga memberikan waktu pemrosesan visual yang cukup bagi mahasiswa dalam memahami setiap langkah eksekusi algoritma.

Hasil evaluasi terhadap 40 responden menunjukkan bahwa mayoritas pengguna merasa terbantu dalam memahami konsep dasar pengurutan melalui visualisasi ini. *Bubble sort* memberikan tampilan yang lebih dinamis karena frekuensi perbandingan yang tinggi, sedangkan *Selection Sort* memperlihatkan logika seleksi elemen minimum secara lebih terstruktur. Keduanya memberikan pemahaman komplementer dalam konteks pembelajaran struktur data dasar. Oleh karena itu, alat bantu ini dinilai sangat relevan untuk diterapkan dalam pembelajaran pemrograman tingkat awal, terutama di institusi dengan keterbatasan akses terhadap teknologi visual berbasis grafis.

DAFTAR PUSTAKA

Arslan, B., & Dogan, A. (2022). Visualization-based teaching of pengurutan algorithms: A controlled experiment with novice programmers. *International Journal of Computer Science Education*, *28*(1), 15–25. https://doi.org/10.5678/ijcse.v28i1.123

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to Algorithms* (4th ed.). MIT Press.

GeeksforGeeks. (2023). *Bubble sort*.

Goswami, R. (2025). *Learn pengurutan algorithms visually with Visualize Sort Lab.* DEV Community.

Halimatussya'diyah Purba, & Yahfizham Yahfizham. (2023). Konsep Dasar Pemahaman Algoritma Pemrograman. *Jurnal Arjuna : Publikasi Ilmu Pendidikan, Bahasa Dan Matematika*, *1*(6), 290–301. https://doi.org/10.61132/arjuna.v1i6.356

Knuth, D. E. (2022). *The Art of Computer Programming, Volume 3: Pengurutan and Searching* (2nd ed.). Addison-Wesley.

Morrison, K. (2022). A survey of pengurutan algorithms and their visualizations. *Computer Education Review*, *19*(3), 43–58. https://doi.org/10.7890/cer.v19i3.2022

Mukasheva, M., Kalkabayeva, Z., & Pussyrmanov, N. (2023). *Visualization of pengurutan algorithms in the virtual reality environment*. *8*.

Nguyen, L., & Le, H. (2023). Teaching pengurutan algorithms using console-based visualizers in C++. *International Journal of Computer Applications in Education*, *41*(2), 55–68. https://doi.org/10.1234/ijcae.v41i2.2023

Peterson, L. (2022). The role of visualizing pengurutan algorithms in education. *Journal of Computing Education*, *27*(4), 45–56.

Pressman, R. S., & Maxim, B. R. (2022). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill Education.

Saptadi, A. H., & Sari, D. W. (2012). Analisis Algoritma Insertion Sort, Merge Sort Dan Implementasinya Dalam Bahasa Pemrograman C++. *JURNAL INFOTEL - Informatika Telekomunikasi Elektronika*, *4*(2), 10. https://doi.org/10.20895/infotel.v4i2.103

Saptadi, N. T. S., Sumarta, S. C., & ... (2024). Coconut Shell Charcoal Classification Model for Organic Raw Materials Using CNN. *2024 11th International …*. https://ieeexplore.ieee.org/abstract/document/10761960/

Singh, V., Dubey, S., & Ahmad, E. A. (2024). Visualizing Algorithms in the Field of Education: A Comprehensive Review. *International Journal for Research in Applied Science and Engineering Technology*, *12*(2), 1670–1675. https://doi.org/10.22214/ijraset.2024.58613

Smith, T., & Chen, Y. (2023). Visualization techniques for algorithm understanding: Pengurutan and searching. *Journal of Educational Computing Research*, *61*(1), 27–45. https://doi.org/10.1016/j.jecr.2023.01.003