Jurnal SIBERNETIKA | Universitas Saintek Muhammadiyah P-ISSN 2355-326X
Vol. 10, No. 2, Desember 2025, hlm. 67 - 77 E-ISSN 2745-5831

IMPLEMENTASI DAN VISUALISASI ALGORITMA BUBBLE SORT DALAM PENYELESAIAN PENUGASAN PEMROGRAMAN MENGGUNAKAN APLIKASI *FLOWGORITHM*

Ardian Fikri Santoso[1], Nurul Anisa Nurdin[2], Naufaldy Muwalat[3], Febriana Nur Hidayah[4], Erzy Maulana Putra[5], Jauhara Hisanah[6]

[1,2,3,4,5,6] Universitas Saintek Muhammadiyah

Email: [1] ardianfikrii06@gmail.com , [2] NurulsNurdins@gmail.com , [3] naufaldy243@gmail.com , [4] febrianahidayah4@gmail.com , [5] ezymaulana21@gmail.com , [6] haraa997@gmail.com.

Abstrak

Pemahaman terhadap algoritma pengurutan dasar seringkali menjadi tantangan bagi mahasiswa tingkat awal dalam mata kuliah Dasar Pemrograman. Algoritma atau metode *Bubble Sort*, meskipun secara konsep sederhana, namun memerlukan visualisasi yang tepat untuk memahami mekanisme pertukaran data secara bertahap. Penelitian ini bertujuan untuk mengimplementasikan dan memvisualisasikan algoritma *Bubble Sort* menggunakan aplikasi *Flowgorithm* sebagai alat bantu dalam penyelesaian penugasan pemrograman.

Metode yang digunakan adalah eksperimental dengan merancang alur logika menggunakan *flowchart* interaktif yang kemudian dikonversi menjadi kode semu (*pseudocode*). Visualisasi dalam *Flowgorithm* memungkinkan pengguna untuk mengamati perubahan indeks dan pergeseran nilai variabel secara real-time melalui fitur *variable watch*.

Hasil penelitian menunjukkan bahwa penggunaan *Flowgorithm* secara signifikan membantu mahasiswa dalam memahami kompleksitas waktu dan logika perulangan bersarang (*nested loop*) yang terdapat pada Bubble Sort. Visualisasi ini menjembatani celah antara konsep teoritis dan praktik pengkodean teknis. Kesimpulan dari penelitian ini menegaskan bahwa pendekatan visual berbasis *flowchart* efektif dalam meningkatkan efisiensi pembelajaran dan membantu mahasiswa dalam menyelesaikan penugasan pemrograman dengan logika yang lebih terstruktur.

Kata Kunci: *Bubble Sort, Flowgorithm, Pemograman, Flowchart*

Abstract

Understanding basic sorting algorithms often poses a challenge for first-year students in Introductory Programming courses. The Bubble Sort algorithm, despite its conceptual simplicity, requires precise visualization to grasp the step-by-step data swapping mechanism. This study aims to implement and visualize the Bubble Sort algorithm using the Flowgorithm application as an instructional tool to assist students in completing programming assignments.

The method employed is experimental, involving the design of logical flows through interactive flowcharts, which are subsequently converted into pseudocode. Visualization within Flowgorithm enables users to observe index changes and variable value shifts in real-time through the variable watch feature.

The results indicate that the utilization of Flowgorithm significantly assists students in comprehending the time complexity and nested loop logic inherent in Bubble Sort. This

visualization effectively bridges the gap between theoretical concepts and technical coding practices. The study concludes that a flowchart-based visual approach is effective in enhancing learning efficiency and guiding students to resolve programming assignments with more structured logic.

Keywords: *Bubble Sort, Flowgorithm, Programming, Flowchart.*

1. PENDAHULUAN

Dasar pemrograman sering kali dianggap sulit oleh pemula karena sifatnya yang abstrak. Salah satu konsep dasar yang wajib dikuasai adalah algoritma pengurutan data (sorting). Pengurutan data memainkan peran yang banyak dipertimbangkan dalam penyelesaian masalah terutama yang berkaitan dengan pengolahan data menjadi lebih baik dan lebih cepat diselesaikan, sehingga menghasilkan data yang lebih akurat.

Pengurutan data atau sorting merupakan salah satu jenis operasi penting dalam pengolahan data[]. Dalam kehidupan sehari-hari hampir setiap saat ditemukan permasalahan-permasalahan yang perlu diselesaikan dengan melakukan operasi pengurutan data. Begitu pentingnya operasi tersebut, sehingga telah banyak dikembangkan metode-metode pengurutan data dan mungkin akan bermunculan metode-metode baru[1].

Pengurutan (sorting) adalah proses penyusunan sekumpulan objek dalam aturan tertentu atau susunan tertentu. Secara umum pengurutan terbagi atas dua jenis yaitu *ascending* (urut naik) dan *descending* (urut turun). Pengurutan data sangat berguna karena data yang terurut akan lebih mudah diperiksa dan diperbaiki jika terdapat kesalahan[2].

Bubble Sort merupakan algoritma pengurutan yang paling dasar yang bekerja dengan cara membandingkan elemen yang berdekatan secara berulang. Namun, tanpa alat bantu visual, mahasiswa sering mengalami kesulitan dalam melacak perubahan data pada setiap iterasi terutama pada penggunaan *nested loop* (perulangan bersarang)[3].

Aplikasi *Flowgorithm* hadir sebagai solusi yang menjembatani antara logika flowchart dan sintaks bahasa pemrograman. Dengan kemampuan visualisasi eksekusi langkah-demi-langkah, *Flowgorithm* memungkinkan pembelajar untuk melihat bagaimana data “mengalir” dan berubah dalam memori secara langsung. Penelitian ini fokus pada bagaimana implementasi Bubble Sort dalam *Flowgorithm* dapat memperjelas logika penugasan pemrograman bagi mahasiswa.

Landasan Teori

Algoritma Bubble Sort

Algoritma merupakan urutan atau struktur yang diterapkan pada bahasa komputer atau pemrograman untuk membantu memecahkan masalah yang datanya tersedia sebagai input dan output sebagai hasil dari proses yang dijalankan. Algoritma pengurutan memiliki kelebihan dan kekurangannya masing masing dan tidak semua algoritma dapat digunakan melakukan pengurutan data.

Algoritma Bubble Sort adalah salah satu dari beberapa jenis sorting yang digunakan untuk mengurutkan data . Cara kerja algoritma ini adalah mengulang proses, melakukan perbandingan antara setiap elemen array dan melakukan penggantian posisi jika urutannya sudah benar. Perbandingan setiap elemen dari array ini berlanjut berjalan sampai kondisi yang ditentukan terpenuhi.

Tahapan-tahapan di dalam algoritma Bubble Sort sebagai berikut: [4]

Langkah pertama

1. Melakukan perbandingan array x[1] dengan array x[2],lalu disusun kembali berdasarkan urutan yang sudah disesuaikan, sehingga x[1] < x[2].
2. Melakukan perbandingan kembali terhadap array x[2] dengan array x[n], lalu disusun kembali berdasarkan urutan yang sudah disesuaikan, sehingga x[2] < x[n].
3. Melakukan perbandingan array x[n-1] dengan array x[n], lalu disusun kembali berdasarkan urutan yang sudah disesuaikan, sehingga array x[n-1] < x[n], setelah (n-1) kali perbandingan, x[n] akan merupakan elemen array terbesar atau terkecil pertama yang sudah terurut.

Langkah kedua

1. Ulangi perbandingan bagian kedua hingga telah membandingkan dan memungkinkan menyusun x[n-2], x[n-1]
2. Setelah elemen array ke (n-2) perbandingan, (n-1) akan merupakan elemen terbesar ke-dua
3. Dan dilanjutkan langkah berikutnya

Langkah ke (n-1)

1. Melakukan perbandingan x[1] dengan x[2] lalu disusun kembali sehingga memunculkan urutan x[1] < x[2]. Sesudah elemen array ke (n-1) langkah, elemen arrayakan tersusun dalam urutan naik ataupun turun sesuai dengan ketentuan yang sudah ditetapkan
2. Dan dilanjutkan langkah berikutnya sampai proses akhir selesai.

2. METODE PENELITIAN

Penelitian ini menggunakan pendekatan deskriptif kualitatif dengan tahap-tahap pengembangan sistem sebagai berikut:

1. Analisis kebutuhan
   Menentukan *input* (array angka) dan *output* (angka terurut).
2. Perancangan *Flowchart*
   - Inisialisasi variabel dan array
   - Penyusunan *nested loop* (perualangan di dalam perulangan)
   - Logika perbandingan (*if statement*) dan teknik penukaran (*swapping*) mengunakan variabel sementara
3. Langkah Pengujian
   Menjalankan program dengan berbagai input data (acak, terurut terbalik, dan sudah terurut) untuk memvalidasi keakuratan algoritma.
4. Observasi Visual
   Memanfaatkan fitur *Step-by-Step Execution* dan *Variable Watcher* untuk mengamati perilaku variabel i dan j serta perubahan posisi elemen dalam array selama proses pengurutan berlangsung.

3. METODE PENGUMPULAN DATA

Dalam penelitian ini, teknik pengumpulan data yang digunakan adalah:

1. Observasi Partisipatif (*Direct Observation*) yaitu mahasiswa atau pengguna berinteraksi dengan *Flowgorithm* saat menyusun logika Bubble Sort.

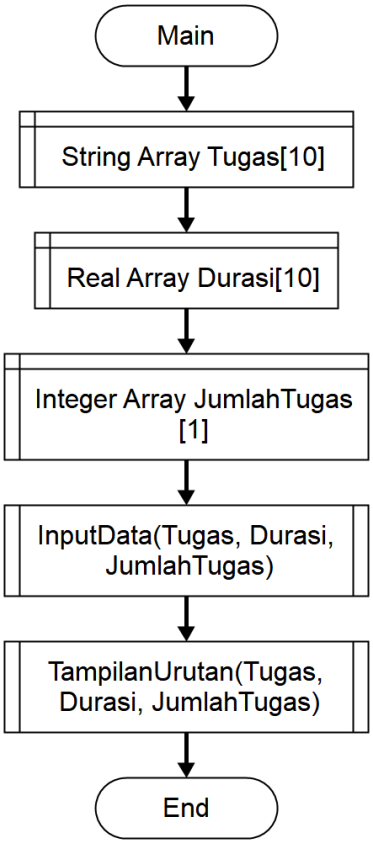
2. Studi Kepustakaan (*Library Research*) yaitu pengumpulan data sekunder mengenai efisiensi algoritma Bubble Sort dan fitur-fitur teknis *Flowgorithm* dari dokumentasi resmi serta penelitian terdahulu yang digunakan sebagai pembanding untuk validasi logika yang dibangun.

4. HASIL DAN PEMBAHASAN

A. Rancangan Kebutuhan Sistem

Dalam pengujian ini maka diperlukan 3 hal, yaitu :
1. *Input* : Array bertipe data real, integer dan string yang masing-masing memiliki maksimal 10 data
2. *Output* : Array akan disusun secara *ascending*
3. *Nested loop* : Digunakan sebagai iterasi berulang untuk melakukan pertukaran data dalam array

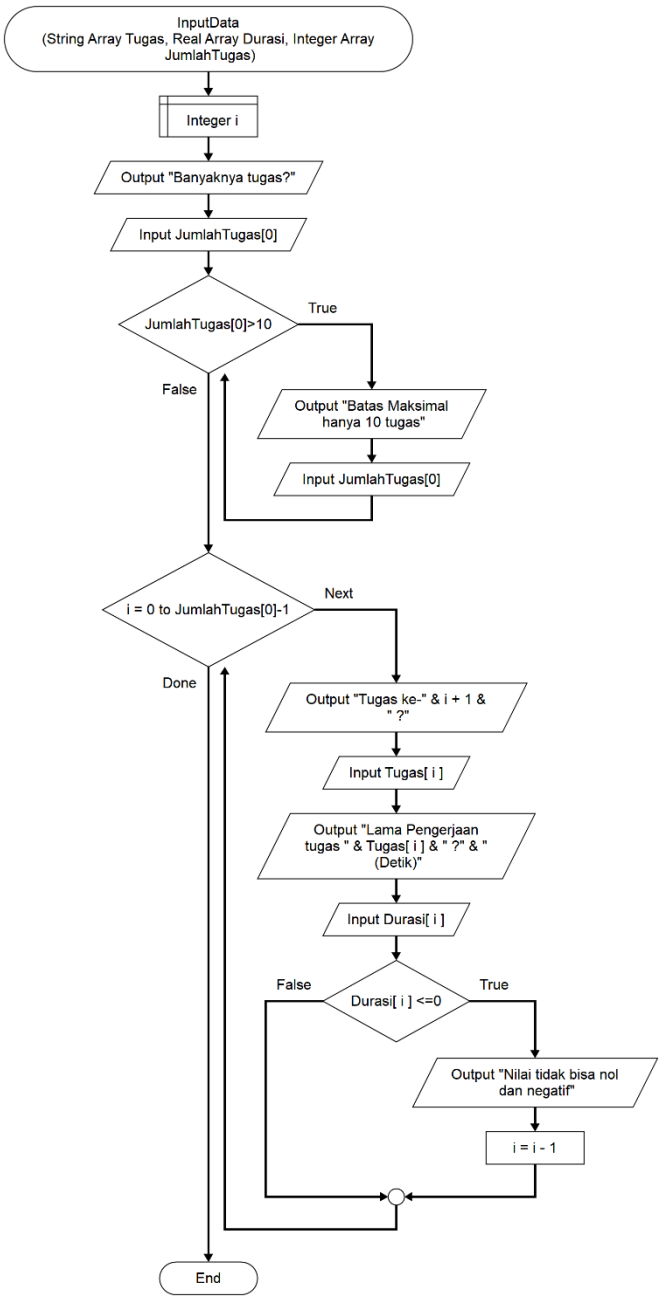
B. Rancangan Sistem



Gambar 1 Fungsi Main Pada *Flowgorithm*

Gambar 1 Menunjukkan tiga array input data untuk pengujian, yaitu array **Tugas**, array **Durasi**, dan array **JumlahTugas**. Array Tugas berfungsi sebagai input data string seperti nama tugas, array Durasi untuk lamanya tugas berlangsung, dan array JumlahTugas untuk mengetahui seberapa banyak data yang akan di input untuk pengujian dengan batas maksimal array tugas dan array durasi sebesar 10 data.

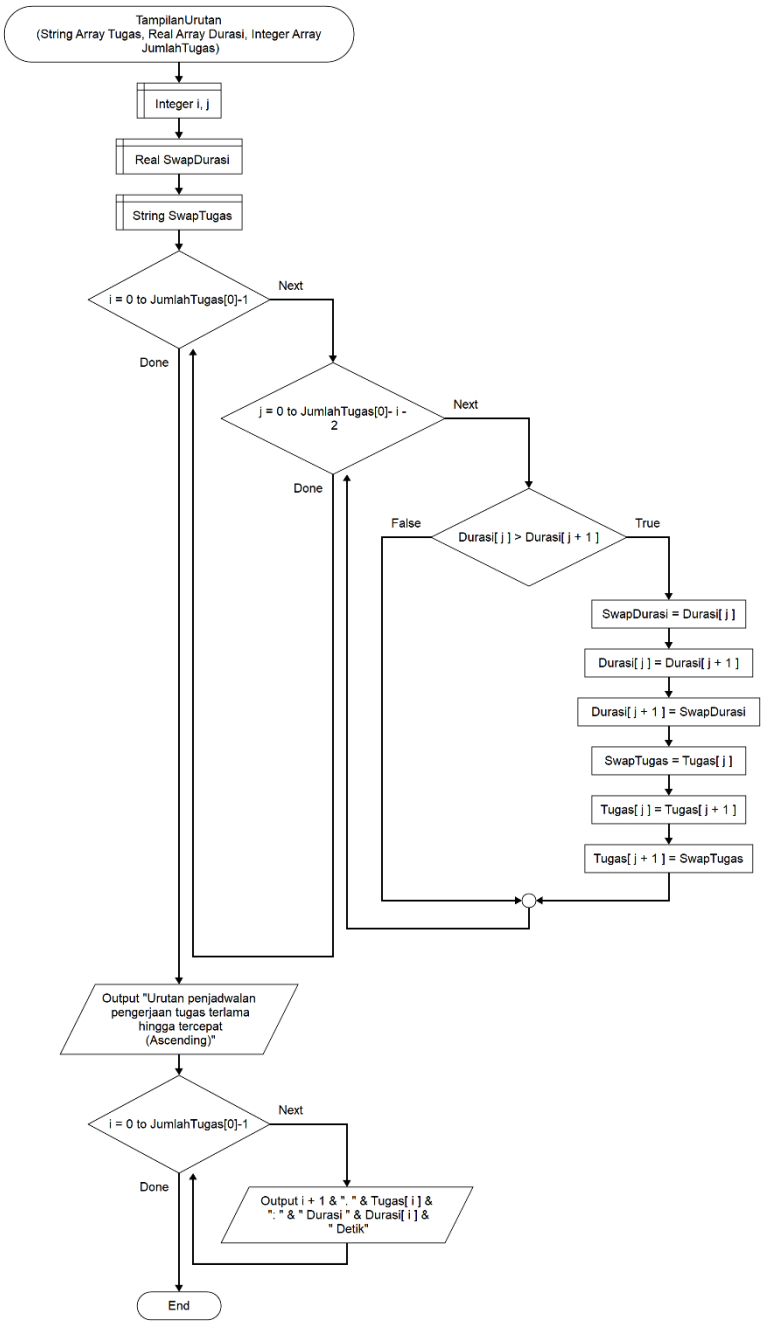
Pada gambar di atas juga terdapat dua fungsi yang berisi program, masing-masing memiliki tugas yang berbeda.



Gambar 2 Fungsi Input Data

Gambar 2 merupakan isi pemrograman pada fungsi InputData yang berfungsi sebagai tempat input data yang akan diuji. Fungsi InputData menggunakan dua *loop for* sederhana. Loop for pertama menggunakan array JumlahTugas yang berfungsi untuk mengetahui berapa banyak data akan di input, serta menghindari penguji untuk memasukkan data lebih dari batas yang telah ditentukan. Untuk loop for kedua bertugas sebagai input data array Tugas dan array

Durasi sesuai dengan jumlah tugas yang telah ditentukan, loop ini juga berisi logika perbandingan untuk menghindari input data 0 dan negatif.



Gambar 3 Fungsi Tampilan Urutan

Gambar 3 adalah isi dari fungsi TampilanUrutan yang berisi algoritma Bubble Sort untuk melakukan sortir data yang diuji serta menampilkan data yang telah tersortir secara terurut menaik (*ascending*). Fungsi TampilanUrutan menggunakan *nested loop* dengan indeks i dan j, logika perbandingan untuk membandingkan array [1] dan array [2] jika array [1] lebih

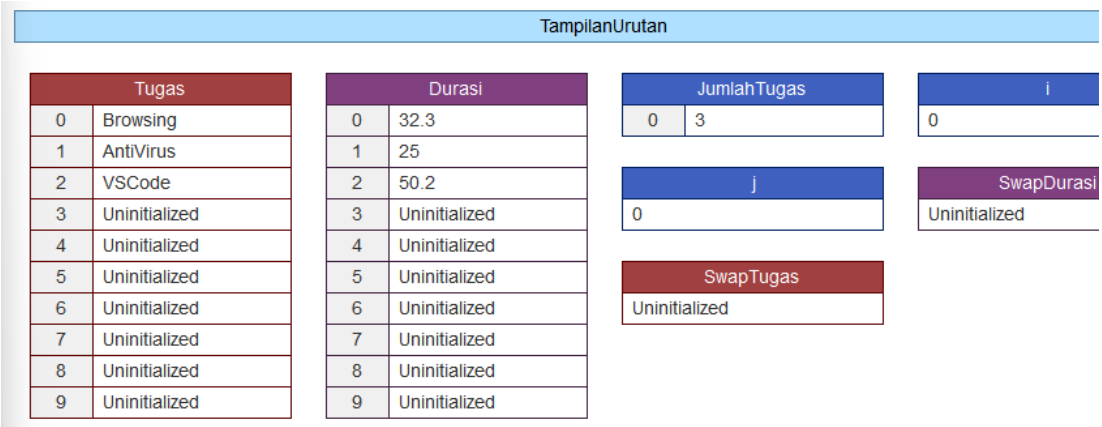
kecil dari array [2] maka akan dilakukan proses *swapping* (sortir) menggunakan variable sementara, yaitu : SwapDurasi dan SwapTugas sebagai tempat penyimpanan data sementara untuk melakukan *swapping* (sortir), Proses ini akan dilakukan terus-menerus hingga seluruh data telah tersortir secara *ascending*.

Setelah semua data sudah tersortir maka loop terakhir akan berjalan untuk menampilkan semua data ke penguji.

C. Implementasi

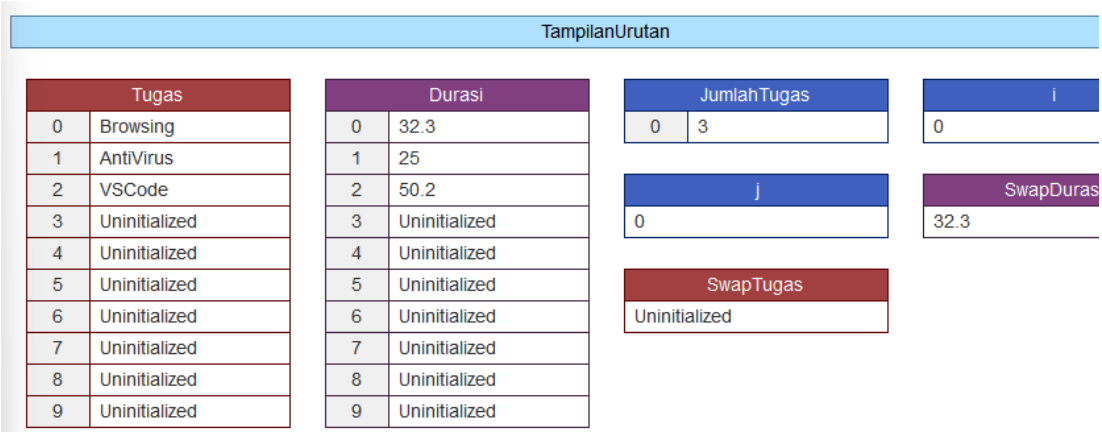
Sesuai dengan rancangan nested loop pada gambar 4.3 implementasi algoritma Bubble Sort dilakukan menggunakan *Flowgorithm* dengan fitur *Step-By-Step Execution* serta observasi visual melalui fitur *Variable Watch*.

Proses *swapping* pada iterasi i=0, j=0 ditunjukkan pada tiga tahapan berikut:



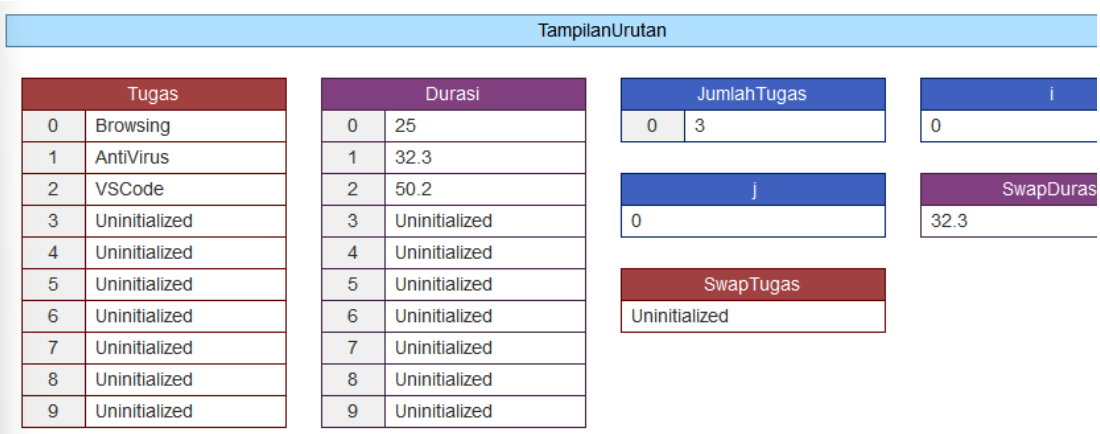
Gambar 4 Kondisi Array Sebelum *Swapping*

Variable Watch menunjukkan array Durasi [0] = 32.3 detik dan Durasi [1] = 25 detik dengan kondisi Durasi[0] > Durasi[1] bernilai True.



Gambar 5 Array Proses *Swapping*

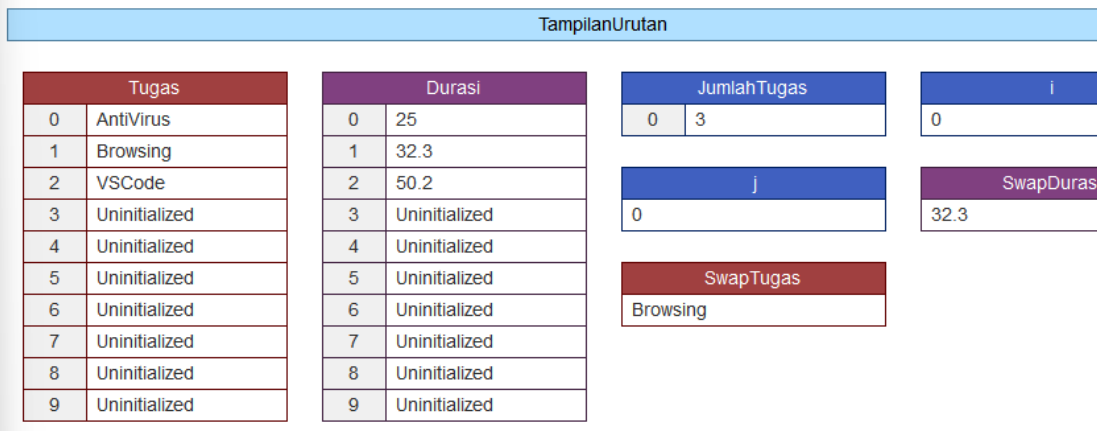
Disaat kondisi logika perbandingan bernilai *true*, maka variable SwapDurasi akan digunakan sebagai tempat pertukaran data sementara, yaitu SwapDurasi akan diisi oleh data dari array Durasi[0] sebelum proses *swapping*.



Gambar 6 Array Setelah *Swapping*

Setelah proses *swapping* kondisi array Durasi[0] dan Durasi[1] berubah menjadi Durasi[0] = 25 detik dan Durasi[1] = 32.3 detik. Array Durasi[1] berubah menjadi array Durasi[1] = 32.3 detik dikarenakan setelah *swapping* data ke Durasi[0] variable SwapDurasi akan mengubah data pada Durasi[1] menjadi data yang sudah dimasukkan sebelum proses *swapping* yaitu data Durasi[0] = 32.3 detik.

Proses tersebut juga akan melakukan hal yang sama kepada array Tugas yaitu melakukan *swapping* data berdasarkan data array Durasi.



Gambar 7 Hasil Proses *Swapping* Array Tugas

Yang akan menghasilkan data array Tugas sesuai dengan data array Durasi. Proses ini akan dilakukan hingga semua jumlah data pada array Tugas dan Durasi tersortir.

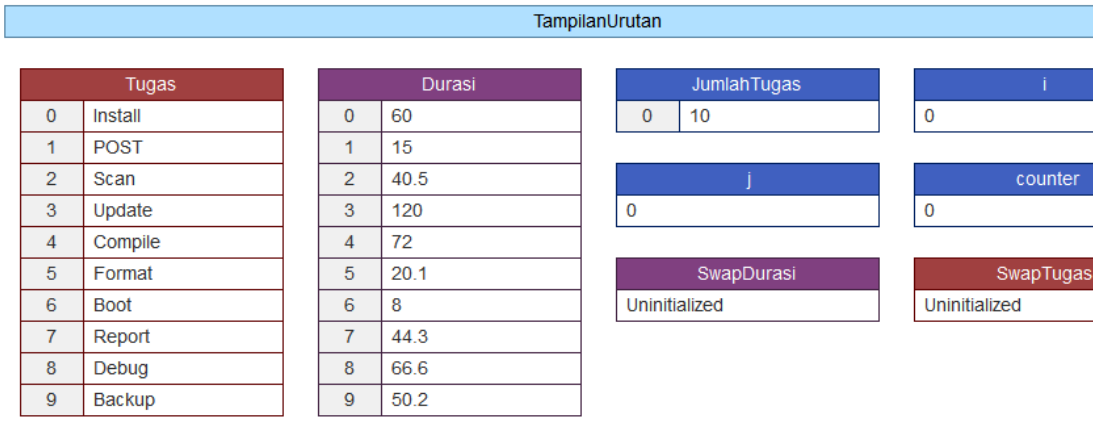
D. Pengujian

Berdasarkan keberhasilan implementasi algoritma Bubble Sort di program *Flowgorithm*, pengujian dilakukan hingga batas maksimal masing-masing array yaitu 10 data dengan tiga skenario input yaitu urutan acak, terbalik, dan urut. Untuk memvalidasi efisiensi dan akurasi algoritma sesuai rancangan sistem.

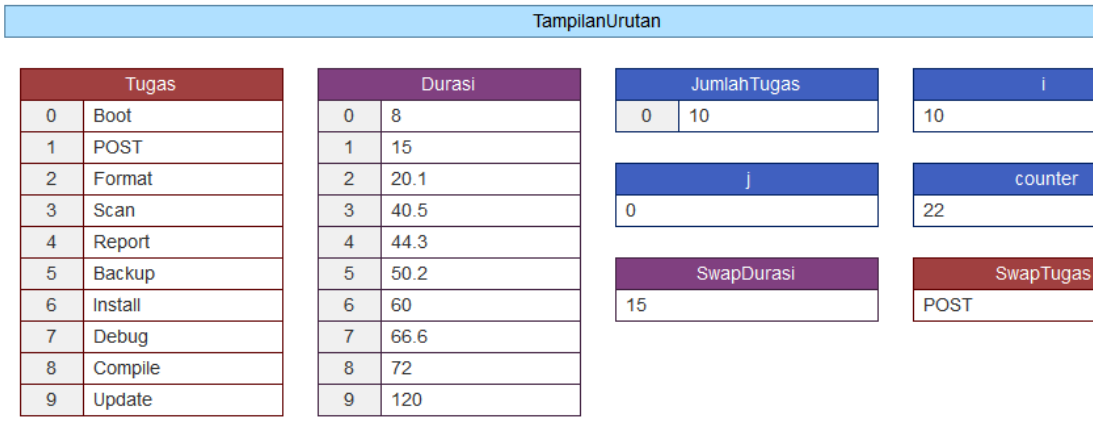
Tabel 1 Tabel Pengujian Algoritma Bubble Sort

Skenario	Ukuran Array	Total *Swapping*	Akurasi
Acak	10 data	22 langkah	100%
Terbalik	10 data	45 langkah	100%
Terurut	10 data	0 langkah	100%

Visualisasi hasil pengujian ditunjukkan pada gambar berikut :

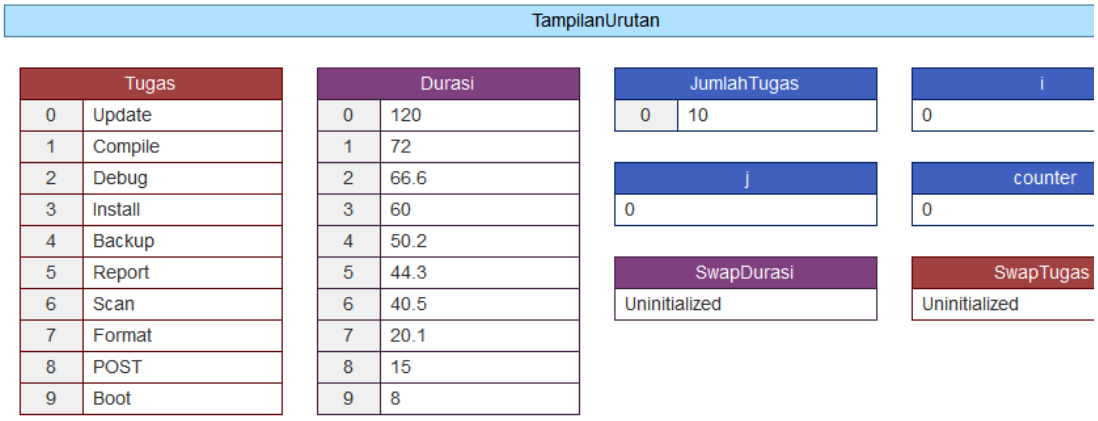


Gambar 8 Skenario Input Acak

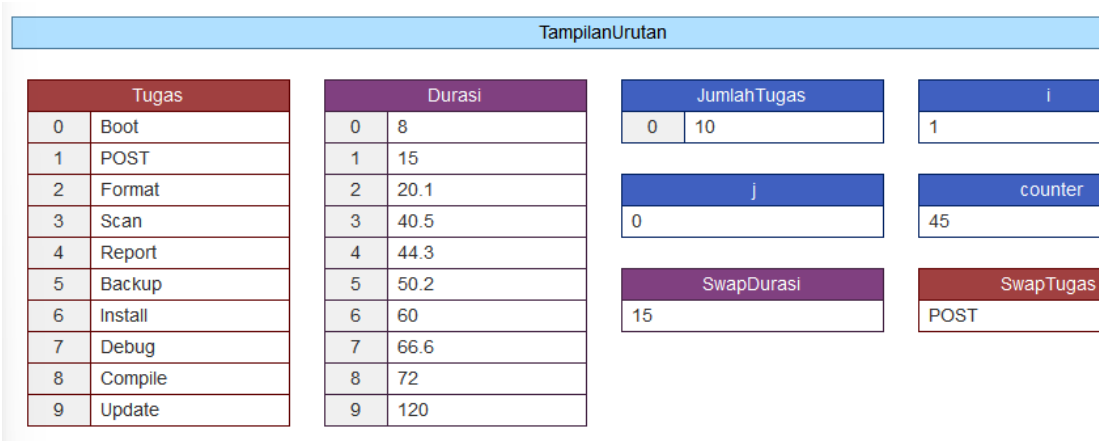


Gambar 9 Hasil Pengujian Input Data Acak

Gambar 8 menunjukkan kondisi array sebelum eksekusi algoritma Bubble Sort dengan urutan acak, sedangkan gambar 9 menunjukkan hasil setelah sorting dengan *swapping* 22 langkah untuk mendapatkan urutan *ascending*.

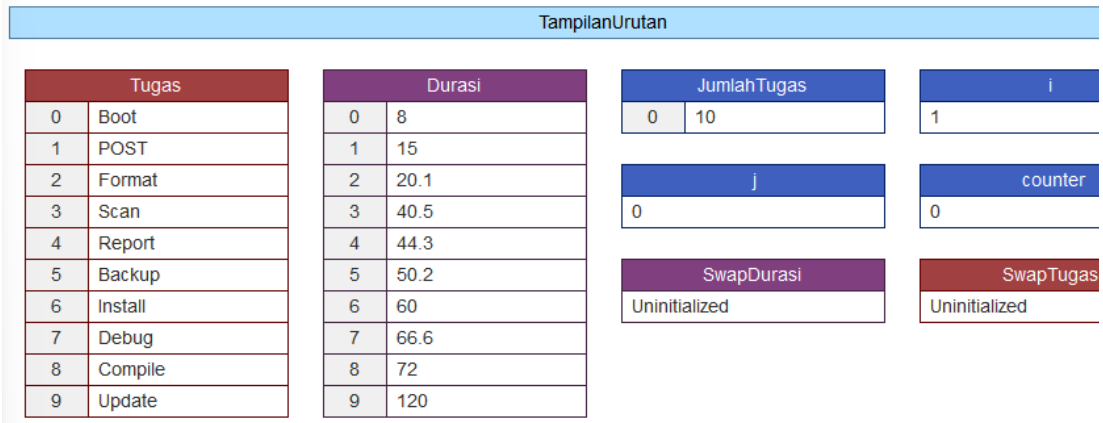


Gambar 10 Skenario Input Data Terbalik



Gambar 11 Hasil Pengujian Input Data Terbalik

Pada gambar 10 Pengujian dilakukan dengan urutan data *descending* atau kondisi terburuk yang dapat dihadapi oleh Bubble Sort, yang dapat ditunjukkan oleh gambar 11 Bubble Sort memerlukan 45 langkah *swapping* untuk mendapatkan urutan *ascending* yang sesuai.



Gambar 12 Hasil Pengujian Input Data Terurut

Gambar 12 menunjukkan tidak adanya perubahan pada array dikarenakan sudah tersusun sesuai yang diperlukan.

5. PENUTUP

Kesimpulan

Berdasarkan hasil implementasi dan pengujian algoritma Bubble Sort menggunakan aplikasi *Flowgorithm*, dapat disimpulkan bahwa:

1. Pendekatan visual berbasis *flowchart* efektif dalam membantu mahasiswa memahami konsep dasar pengurutan data. Proses visualisasi melalui fitur *Step-by-Step Execution* dan *Variable Watch* memungkinkan pengguna mengamati secara langsung mekanisme *nested loop*, proses perbandingan, serta teknik *swapping* antar elemen array.
2. Hasil pengujian terhadap tiga skenario input (acak, terbalik, dan terurut) menunjukkan tingkat akurasi 100%, dengan jumlah langkah *swapping* yang berbeda sesuai kondisi data. Pada kondisi terbalik (*worst case*), jumlah *swapping* lebih banyak dibandingkan kondisi acak dan terurut. Hal ini membantu mahasiswa memahami konsep kompleksitas waktu pada algoritma Bubble Sort secara lebih konkret. Dengan demikian, penggunaan *Flowgorithm* mampu menjembatani kesenjangan antara pemahaman teori algoritma dan praktik implementasi pemrograman, serta meningkatkan kemampuan mahasiswa dalam menyusun logika program secara sistematis dan terstruktur.
3. Selain itu, pemanfaatan fitur *variable watch* terbukti efektif dalam meminimalkan kesalahan logika (*logic error*) yang sering dialami pemula saat melacak perubahan nilai variabel sementara selama proses pertukaran data. Visualisasi interaktif ini secara nyata mengurangi sifat abstrak dari pemrograman dasar, sehingga mahasiswa dapat lebih fokus pada penguatan alur logika sebelum berhadapan dengan kompleksitas sintaksis bahasa pemrograman tertentu. Dengan demikian, penggunaan *Flowgorithm* mampu menjembatani kesenjangan antara pemahaman teori algoritma dan praktik implementasi pemrograman, serta meningkatkan kemampuan mahasiswa dalam menyusun logika program secara sistematis dan terstruktur.

Saran

Berdasarkan hasil penelitian yang telah dilakukan, beberapa saran yang dapat diberikan adalah sebagai berikut:

1. Penggunaan *Flowgorithm* disarankan untuk diterapkan secara lebih luas dalam pembelajaran Dasar Pemrograman, khususnya pada materi algoritma dan struktur kontrol perulangan.
2. Penelitian selanjutnya dapat melakukan perbandingan visualisasi dengan algoritma sorting lainnya seperti Selection Sort, Insertion Sort, atau Quick Sort untuk menganalisis perbedaan kompleksitas dan efisiensi.

DAFTAR PUSTAKA

[1] H. Triansyah, “Implemetasi Metode Bubble Sort Dalam Pengurutan Indeks Prestasi Mahasiswa,” *J. Ilm. Inform.*, vol. 7, no. 01, pp. 48–53, 2019.
[2] N. Mahrozi and M. Faisal, “Analisis perbandingan kecepatan algoritma selection sort dan bubble sort,” *Sci. J. Ilm. Sains dan Teknol.*, vol. 1, no. 2, pp. 89–98, 2023.
[3] R. Nasution, A. Syahputra, A. Widiyanto, D. Subuhanto, and A. Y. Abdillah, “Persepsi Mahasiswa Informatika Terhadap Keefektifan Algoritma Bubble Sort dalam Mengurutkan Data,” *Blend Sains J. Tek.*, vol. 1, no. 3, pp. 220–225, 2023.
[4] M. F. Abdullah, I. Hafiza, R. Wahyuni, and A. Syahputra, “Penggunaan Algoritma

Bubble Sort dalam Pengurutan Nomor Induk Mahasiswa," *Hello World J. Ilmu Komput.*, vol. 2, no. 1, pp. 14–19, 2023.

[5] A. B. Panggabean, R. R. Htb, I. Perina, Y. L. Toro, and A. Syahputra, "Implementasi Algoritma Bubble Sort pada Sistem Pelayanan Perpustakaan Menggunakan Laravel," Sudo J. Tek. Inform., vol. 2, no. 1, pp. 19–27, 2023.

[6] I. Gunawan, S. Sumarno, and H. S. Tambunan, "Penggunaan Algoritma Sorting Bubble Sort Untuk Penentuan Nilai Prestasi Siswa," Sistemasi: Jurnal Sistem Informasi, vol. 8, no. 2, pp. 296–304, 2019.

[7] Y. Astuti, "Analisis Pengujian Data Algoritma Bubble Sort," REMIK: Riset dan E-Jurnal Manajemen Informatika Komputer, vol. 7, no. 3, pp. 1413–1420, 2023.