

Aplikasi Konsep Algoritma Greedy pada Pengembalian Pembelian

Nabila Muthia Putri 120140023 (Author)

Program Studi Teknik Informatika

Jurusan Teknologi Produksi dan Industri

Institut Teknologi Sumatera, Jalan Terusan Ryacudu Lampung Selatan

E-mail: nabila.120140023@student.itera.ac.id

Abstrak—Pada zaman modern ini, diperlukan optimalisasi dalam menyelesaikan permasalahan untuk memanfaatkan waktu maupun tenaga sebaik mungkin, salah satunya adalah permasalahan dalam pengembalian uang belanja di suatu toko. Biasanya, toko membutuhkan seorang kasir untuk mengembalikan uang belanja secara manual, namun permasalahan ini dapat diselesaikan dengan praktis menggunakan bantuan teknologi. Dengan menggunakan algoritma *greedy*, dapat diperoleh nominal kembalian uang belanja secara otomatis. Penulisan makalah ini bertujuan untuk menggagaskan modernisasi kegiatan transaksi luring di toko untuk mendukung kemudahan dalam jual-beli. Hasil dari penulisan makalah ini menunjukkan bahwa penggunaan teknologi untuk menyelesaikan permasalahan transaksi dapat memudahkan pihak pemilik toko dan pembeli jika tidak ada kesalahan pada program yang digunakan untuk implementasi.

Kata kunci—kembalian; otomatis; *greedy*; optimal

I. PENDAHULUAN

Pada abad ke-21 ini, terdapat terobosan baru dalam kegiatan jual-beli, salah satunya adalah konsep *cashierless* dari Amazon Go yang memungkinkan pelanggan untuk memasuki toko dengan memindai aplikasi Amazon Go dan langsung keluar tanpa harus berdiri di antrean kasir. Namun, toko-toko di Indonesia bahkan belum menerapkan konsep *self-checkout* yang memungkinkan pelanggan untuk menghitung jumlah pembelian dengan cara memindai produk yang dibeli, kemudian mengkalkulasi jumlah pembelian serta uang yang dibayar, kemudian mendapat kembalian yang sesuai.

Maka dari itu, terdapat mesin *self-checkout* (SCO) yang dapat digunakan untuk mendukung kepraktisan dalam jual-beli, yang menggunakan konsep dari algoritma *greedy* dalam mengkalkulasi jumlah nominal pengembalian. Algoritma *greedy* bekerja secara *step-by-step* dalam memilih langkah-langkah yang memberikan keuntungan. Algoritma *greedy* memilih solusi optimum lokal, tanpa memikirkan konsekuensi pada langkah ke depannya. Algoritma *greedy* tidak selalu mengarah pada solusi global yang optimal, karena tidak mempertimbangkan seluruh data yang ada layaknya *exhaustive search* dari algoritma *Brute Force*.

Konsep dari mesin SCO memiliki beberapa kriteria yang mirip dengan konsep *vending machine* yang saat ini sudah dapat ditemui di beberapa tempat tertentu di Indonesia. Kemiripannya adalah kedua mesin tersebut berjalan secara

mandiri, dan dapat mendeteksi harga barang serta nominal uang yang dimasukkan oleh pelanggan. [1]

II. TEORI DASAR

A. Algoritma Greedy

Algoritma *greedy* bekerja secara *step-by-step* dalam memilih langkah-langkah yang memberikan keuntungan. Algoritma *greedy* memilih solusi optimum lokal, tanpa memikirkan konsekuensi pada langkah ke depannya. Algoritma *greedy* tidak selalu mengarah pada solusi global yang optimal, karena tidak mempertimbangkan seluruh data yang ada layaknya *exhaustive search* dari algoritma *Brute Force*. Solusi yang diberikan oleh pendekatan algoritma *greedy* tidak mempertimbangkan data dan pilihan untuk kemungkinan ke depan. Dalam beberapa kasus, algoritma *greedy* membuat keputusan yang pada saat itu memberikan solusi terbaik, tetapi dalam beberapa kasus lainnya mungkin tidak berlaku demikian.

Pendekatan dengan algoritma *greedy* memiliki beberapa pengorbanan untuk menemukan solusi optimal pada setiap permasalahan. Dalam permasalahan *activity selection*, jika lebih banyak aktivitas dapat dilakukan sebelum menyelesaikan aktivitas yang dikerjakan saat ini, aktivitas ini dapat dilakukan dalam waktu yang sama. Alasan lainnya adalah untuk membagi masalah secara rekursif berdasarkan suatu kondisi, tanpa perlu menggabungkan semua solusi. Dalam permasalahan *activity selection*, langkah *recursive division* dicapai dengan memindai daftar item hanya sekali kemudian mempertimbangkan aktivitas tertentu.

Solusi optimal global dapat diperoleh dengan membuat solusi optimal lokal dari algoritma *greedy*. Solusi yang diberikan oleh algoritma *greedy* mungkin bergantung pada pilihan sebelumnya, tetapi tidak bergantung pada langkah ke depannya. Hal tersebut secara berulang membuat satu pilihan *greedy* kemudian mengurangi permasalahan yang diberikan sehingga terbagi menjadi permasalahan yang lebih kecil.

Suatu permasalahan menunjukkan substruktur optimal jika solusi optimal dari masalah tersebut berisi solusi optimal untuk submasalah. Hal itu berarti bahwa submasalah dapat dipecahkan dan membangun solusi untuk pemecahan masalah yang lebih besar.

Karakteristik pendekatan *greedy* antara lain memiliki list yang teratur dari data (bisa berupa keuntungan, biaya, nilai,

dll.), dan memiliki nilai maksimum dari semua data (keuntungan maksimal, nilai maksimal, dll.).

Keuntungan dalam menggunakan pendekatan algoritma *greedy* antara lain mudah diterapkan, biasanya memiliki kompleksitas waktu yang lebih sedikit, algoritma *greedy* dapat digunakan untuk tujuan optimasi. Namun di sisi lain, terdapat kekurangan ketika menggunakan pendekatan algoritma *greedy* yaitu solusi optimal lokal mungkin tidak selalu optimal global. [2]

B. Convenience Store di Indonesia

Industri *convenience store* di Indonesia adalah salah satu peluang bisnis yang paling menguntungkan di negara ini. Banyak pemilik bisnis memasuki industri ini melalui penggunaan waralaba, tetapi beberapa pemilik bisnis memilih untuk memasuki industri ini setelah memulai *convenience store* milik mereka sendiri sejak awal.

Industri *convenience store* tumbuh cepat di Indonesia karena keberadaan *convenience store* internasional agak berkurang mengingat sebagian besar penduduk lokal lebih menyukai *convenience store* domestik. Toko yang paling populer adalah Indomaret dan Alfamart. Kedua toko ini tersebar luas di seluruh Indonesia dengan ribuan outlet yang tersebar di seluruh tanah air. Popularitas mereka sangat didasari dengan harga dan kenyamanan yang bagi para pelanggannya. Toko-toko ini juga menyediakan Wi-Fi gratis serta AC. Namun, Indomaret dan Alfamart bukan satu-satunya toko serba ada yang ada di industri *convenience store* di Indonesia.

Convenience store dapat mempertimbangkan perubahan kebutuhan dan preferensi konsumen. Toko-toko ini memantau dengan cermat tanggapan konsumen terhadap produk yang mereka simpan, kemudian akan membuat perubahan jika dirasa perlu. *Convenience store* juga menerapkan standar kebersihan yang tinggi dan cukup memberi atensi penuh mengenai keamanan pangan. Tidak seperti toko lain, *convenience store* menyediakan ruang yang cukup bagi para pelanggan untuk memungkinkan mereka dapat memilih barang yang diinginkan secara gratis dan cepat. Maka dari itu, sesuai dengan namanya, konsep *convenience store* dapat menghemat waktu lebih banyak. Kehadiran kasir yang banyak juga membantu dalam mempercepat proses transaksi. Banyak *convenience store* di Indonesia juga menawarkan kemudahan finansial karena banyak bank-bank telah memasang anjungan tunai mandiri (ATM) di *convenience store*. Kemudahan akses uang tunai ini dapat memudahkan konsumen saat berbelanja.

Keuntungan lain dari *convenience store* adalah pihak toko memiliki hubungan pelanggan yang positif. Maka dari itu, kepentingan pelanggan akan selalu didahulukan. Hal ini telah memungkinkan pemilik bisnis untuk mempertahankan sejumlah besar pelanggan sehingga meningkatkan kredibilitas toko di pasar. Rutinitas operasional 24 jam yang diterapkan di *convenience store* juga memungkinkan konsumen untuk dapat masuk kapan saja untuk berbelanja. Penerapan konsep ini sangat bermanfaat untuk banyak turis dari dalam maupun dari luar negeri. [3]

C. Mesin Self-Checkout (SCO)

Banyak orang telah melihat tempat pembayaran di *convenience store*. Namun untuk memanfaatkan kemajuan teknologi serta memaksimalkan kepraktisan dalam kegiatan jual-beli, dapat menggunakan sebuah alternatif untuk jalur *checkout* yang sebelumnya biasa dikelola oleh kasir. Terobosan ini diperkenalkan untuk menawarkan kenyamanan yang lebih tinggi serta menyediakan opsi *checkout* yang lebih cepat kepada para pelanggan. [4]



Gambar 2.1 Source: www.darwinmag.com

Sebagian besar mesin *self-checkout* memiliki fungsi dasar seperti berikut:

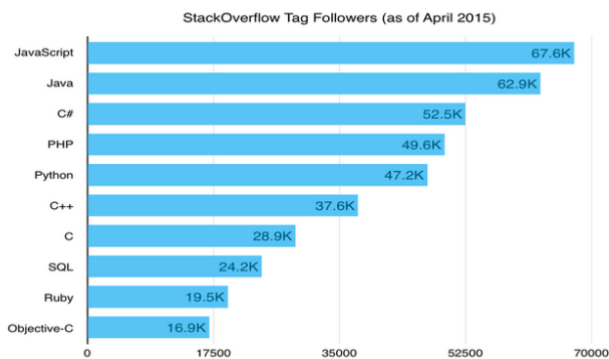
1. Lampu lajur yang dapat digunakan untuk panggilan petugas toko,
2. Monitor yang dapat digunakan dengan teknologi layar sentuh,
3. Dudukan untuk meletakkan keranjang belanja,
4. Pemindai *barcode* dan timbangan berat dari produk yang dibeli,
5. Bagian untuk melakukan pembayaran,
6. Bagian untuk memasukkan PIN ATM

D. Bahasa Pemrograman C++

C++ merupakan bahasa pemrograman yang memiliki tujuan yang bersifat umum untuk dapat digunakan untuk mengembangkan sistem operasi, browser, game, dan sebagainya. C++ mendukung berbagai cara pemrograman seperti prosedural, berorientasi objek, fungsional, dan sebagainya. Hal-hal tersebut membuat C++ menjadi bahasa pemrograman yang kedudukannya kuat dan juga fleksibel. [5]

Bahasa pemrograman C++ dibuat oleh Bjarne Stroustrup di Bell Labs sekitar tahun 1980. C++ sangat mirip dengan C yang ditemukan oleh Dennis Ritchie pada awal 1970-an. Maka dari itu, C++ sangat kompatibel dengan C sehingga mungkin akan mengkompilasi lebih dari 99% program C tanpa mengubah baris dari *source code*. C++ adalah bahasa yang terstruktur dengan baik dan lebih aman daripada C karena berbasis berorientasi objek.

Beberapa bahasa komputer ditulis untuk tujuan tertentu masing-masing. Misalnya, Java yang awalnya dirancang untuk mengontrol pemanggang roti dan beberapa elektronik lainnya. C yang awalnya dikembangkan untuk pemrograman sistem operasi. Pascal yang dikonsepkan untuk mengajarkan teknik pemrograman yang tepat. Namun C++ adalah dengan bahasa tujuan yang bersifat umum.



Gambar 2.2 Source: stackoverflow.com

Grafik di atas menunjukkan bahwa C++ dapat bersaing di antara bahasa pemrograman lainnya jika dilihat dari kepopulerannya walaupun tergolong bahasa pemrograman kuno dibanding bahasa pemrograman lainnya yang lebih modern.

Terdapat lima konsep dasar dari C++, antara lain:

1. Variabel

Variabel diibaratkan sebagai tulang punggung dari setiap bahasa pemrograman. Variabel merupakan cara untuk menyimpan beberapa informasi yang akan digunakan setelahnya. Setelah dideklarasikan dan didefinisikan, variabel dapat digunakan berkali-kali dalam lingkup tempat di mana variabel tersebut dideklarasikan.

2. Struktur Kontrol

Ketika sebuah program dijalankan, kode dibaca oleh kompiler baris demi baris dari atas ke bawah, dan sebagian besar dari kiri ke kanan. Ketika kode sedang

dibaca dari atas ke bawah, mungkin akan menemui titik di mana kompiler harus membuat keputusan. Berdasarkan keputusan tersebut, program dapat melompat ke bagian kode yang berbeda. Bahkan mungkin membuat kompiler menjalankan kembali bagian tertentu, atau dapat melewati banyak kode. Sebuah program komputer memiliki seperangkat aturan ketat untuk memutuskan aliran eksekusi program.

3. Struktur Data

Struktur data adalah cara yang bagus untuk mengaplikasikan dalam membuat ribuan variabel. C++ berisi banyak jenis struktur data bawaan, yang paling sering digunakan adalah array yang akan digunakan pada pengimplementasian ini.

4. Sintaks

Sintaksnya dari bahasa pemrograman C++ adalah tata letak kata, ekspresi, dan simbol. Sintaks adalah beberapa aturan yang terdefinisi yang memungkinkan *programmer* untuk membuat beberapa bagian dari perangkat lunak untuk berfungsi dengan baik. Tapi, jika *programmer* tidak mematuhi aturan bahasa pemrograman atau sintaks, maka program akan menyatakan kesalahan.

5. Tools

Di dunia nyata, *tools* adalah sesuatu yang bersifat fisik untuk membantu seseorang dalam menyelesaikan pekerjaan tertentu. Namun *tools* dalam pemrograman adalah beberapa bagian dari perangkat lunak yang bila digunakan dengan kode dapat memungkinkan *programmer* untuk memprogram lebih cepat. *Tools* yang dianggap paling penting oleh banyak orang adalah IDE (*Integrated Development Environment*). IDE memastikan bahwa file dan folder diatur dan memberi cara yang praktis untuk membukanya. [6]

III. ANALISIS DAN PEMBAHASAN

A. Header

<bits/stdc++.h> adalah file *header*. File ini mencakup semua perpustakaan (*library*) standar. Terkadang dalam beberapa kompetisi pemrograman, ketika peserta *programmer* harus menghemat waktu saat menyelesaikan permasalahan, maka menggunakan file *header* ini sangat membantu. [7]

Menggunakan file *header* ini akan menyertakan banyak file, yang kadang-kadang sebenarnya tidak diperlukan dalam program. Hal tersebut dapat meningkatkan waktu kompilasi dan ukuran program yang ditulis. Beberapa kelemahan besar dari file *header* ini antara lain:

1. <bits/stdc++.h> bukan file *header* standar dari *library* GNU C++. Maka dari itu, beberapa kompiler mungkin gagal mengkompilasi *source code* yang menggunakan file *header* ini.

2. Dengan menggunakan file *header* ini, mungkin memerlukan waktu lebih lama untuk proses kompilasi yang sebenarnya tidak begitu diperlukan.
3. Karena file *header* ini bukan merupakan bagian dari *library* standar dari C++, sehingga penggunaan file *header* ini bersifat tidak *portable*.
4. Dalam penggunaan file *header* ini, compiler selalu mencoba mengimpor *header* secara rekursif setiap kali kode dikompilasi.

B. Array

Array adalah serangkaian elemen dengan tipe yang sama yang ditempatkan di lokasi memori yang berdekatan dan dapat direferensikan secara individual dengan menambahkan indeks.

Misalnya, lima nilai bertipe integer dapat dideklarasikan sebagai array tanpa harus mendeklarasikan 5 variabel berbeda dengan masing-masing pengenalnya sendiri. Sebagai gantinya, dapat menggunakan array yang menampung lima nilai integer di lokasi memori yang berdekatan, dan kelimanya dapat diakses menggunakan *identifier* yang sama, dengan indeks yang tepat.

Seperti layaknya variabel biasa, array harus dideklarasikan sebelum digunakan. Deklarasi khas untuk array di C++ adalah: nama[jumlah elemen]; dengan tipe data yang valid (seperti int ataupun float), nama adalah *identifier* yang valid, dan jumlah elemen yang selalu diapit tanda kurung siku [] menentukan panjang array.

Elemen dalam larik bisa diinisialisasi secara eksplisit ke nilai tertentu saat dideklarasikan, dengan menyertakan nilai awal tersebut dalam kurung kurawal {}.

Nilai dari salah satu elemen dalam array dapat diakses seperti nilai variabel reguler dengan tipe yang sama. Sintaks yang digunakan adalah: nama[indeks];. [8]

C. Vector

Vektor merupakan array dinamis dengan kemampuan untuk mengubah ukurannya sendiri secara otomatis ketika elemen dimasukkan atau dihapus, dengan penyimpanan yang ditangani secara otomatis. Elemen vektor ditempatkan dalam penyimpanan yang berdekatan sehingga dapat diakses dan dilalui menggunakan iterator. Dalam vektor, data disisipkan di akhir sehingga membutuhkan waktu yang berbeda, karena kadang-kadang mungkin ada penambahan data untuk memperluas cakupan array. [9]

Fungsi-fungsi tertentu yang terkait dengan vektor adalah:

1. Iterator

- `begin()` berfungsi untuk mengembalikan iterator yang menunjuk ke elemen pertama dalam vektor
- `end()` berfungsi untuk mengembalikan iterator yang menunjuk ke elemen yang mengikuti elemen terakhir dalam vektor

- `rbegin()` berfungsi untuk mengembalikan iterator terbalik yang menunjuk ke elemen terakhir dalam vektor dari elemen terakhir ke elemen pertama
- `rend()` berfungsi untuk mengembalikan iterator terbalik yang menunjuk ke elemen sebelum elemen pertama dalam vektor
- `cbegin()` berfungsi untuk mengembalikan iterator konstan yang menunjuk ke elemen pertama dalam vektor
- `cend()` berfungsi untuk mengembalikan iterator konstan yang menunjuk ke elemen yang mengikuti elemen terakhir dalam vektor
- `crbegin()` berfungsi untuk mengembalikan iterator terbalik konstan yang menunjuk ke elemen terakhir dalam vektor dari elemen terakhir ke elemen pertama
- `crend()` berfungsi untuk mengembalikan iterator terbalik konstan yang menunjuk ke elemen sebelum elemen pertama dalam vektor

2. Kapasitas

- `size()` berfungsi untuk mengembalikan jumlah elemen dalam vektor
- `max_size()` berfungsi untuk mengembalikan jumlah maksimum elemen yang dapat ditampung oleh vektor
- `capacity()` berfungsi untuk mengembalikan ukuran ruang penyimpanan yang saat ini dialokasikan ke vektor yang dinyatakan sebagai jumlah elemen
- `resize(n)` berfungsi untuk mengubah ukuran penyimpanan sehingga berisi elemen 'n'
- `empty()` berfungsi untuk mengembalikan pernyataan apakah penyimpanan kosong
- `shrink_to_fit()` berfungsi untuk mengurangi kapasitas penyimpanan agar sesuai dengan ukurannya dan menghancurkan semua elemen di luar kapasitas
- `reserve()` berfungsi untuk meminta kapasitas vektor setidaknya cukup untuk memuat n elemen

3. Akses elemen

- *reference operator* `[g]` berfungsi untuk mengembalikan referensi ke elemen pada posisi 'g' dalam vektor
- `at(g)` berfungsi untuk mengembalikan referensi ke elemen pada posisi 'g' dalam vektor
- `front()` berfungsi untuk mengembalikan referensi ke elemen pertama dalam vektor
- `back()` berfungsi untuk mengembalikan referensi ke elemen terakhir dalam vektor

- `data()` berfungsi untuk mengembalikan penunjuk langsung ke larik memori yang digunakan secara internal oleh vektor untuk menyimpan elemen miliknya
4. Pengubah
- `assign()` berfungsi untuk memberikan nilai baru ke elemen vektor dengan mengganti yang lama
 - `push_back()` berfungsi untuk mendorong elemen ke dalam vektor dari belakang
 - `pop_back()` berfungsi untuk memunculkan atau menghapus elemen dari vektor dari belakang
 - `insert()` berfungsi untuk memasukkan elemen baru sebelum elemen pada posisi yang ditentukan
 - `erase()` berfungsi untuk menghapus elemen dari wadah dari posisi atau rentang yang ditentukan
 - `swap()` berfungsi untuk menukar isi dari satu vektor dengan vektor lain dengan tipe yang sama namun dengan ukuran yang mungkin berbeda
 - `clear()` berfungsi untuk menghapus semua elemen penyimpanan vektor
 - `emplace()` berfungsi untuk memperluas penyimpanan dengan memasukkan elemen baru di posisi
 - `emplace_back()` berfungsi untuk memasukkan elemen baru ke dalam penyimpanan vektor yang ditambahkan ke akhir vektor

IV. IMPLEMENTASI

Permasalahan yang diambil untuk pengimplementasian algoritma *greedy* ini adalah menghitung jumlah kembalian dari uang belanja yang dimasukkan oleh pelanggan sesuai dengan jumlah total pembelanjaan dengan cara mengembalikan nominal yang berlaku dari mata uang di Indonesia, antara lain: uang koin 100, uang koin 200, uang koin 500, uang koin 1000, uang kertas 2000, uang kertas 5000, uang kertas 10000, uang kertas 20000, uang kertas 50000, dan uang kertas 100000.

Kemudian ketika pelanggan memindai produk yang mereka beli di mesin *self-checkout*, program akan mengkalkulasi jumlah total pembelanjaan. Setelah itu, ketika pelanggan memasukkan uang, mesin *self-checkout* akan mendeteksi nominal uang secara otomatis. Setelah itu, program akan mengkalkulasi selisih antara uang yang dimasukkan pelanggan dengan jumlah total pembelanjaan yang menghasilkan jumlah kembalian. Dengan algoritma *greedy*, dapat diperoleh nominal uang yang sesuai dengan pengembalian.

Berikut ini contoh output program:

```
Jumlah belanja (Rp.): 170000
Masukkan uang yang dibayar (Rp.): 200000
Kembalian berjumlah (Rp.): 30000
Uang yang dikembalikan adalah (Rp.): 20000 10000
```

Dalam percobaan di atas, jumlah total pembelanjaan dan uang yang dimasukkan oleh pelanggan merupakan pecahan

uang kertas bulat yang kembaliannya juga merupakan pecahan uang kertas. Maka dari itu, untuk setiap nominal pengembaliannya juga merupakan pecahan uang kertas.

```
Jumlah belanja (Rp.): 65500
Masukkan uang yang dibayar (Rp.): 70000
Kembalian berjumlah (Rp.): 4500
Uang yang dikembalikan adalah (Rp.): 2000 2000 500
```

Pada percobaan di atas, jumlah total pembelanjaan merupakan uang kertas dan uang koin, dan uang yang dimasukkan oleh pelanggan merupakan uang kertas. Maka dari itu, untuk setiap nominal pengembaliannya adalah uang kertas dan uang koin.

```
Jumlah belanja (Rp.): 105700
Masukkan uang yang dibayar (Rp.): 120500
Kembalian berjumlah (Rp.): 14800
Uang yang dikembalikan adalah (Rp.): 10000 2000 2000 500 200 100
```

Pada percobaan di atas, jumlah total pembelanjaan merupakan uang kertas dan uang koin, dan uang yang dimasukkan oleh pelanggan merupakan uang kertas dan uang koin. Maka dari itu, untuk setiap nominal pengembaliannya adalah uang kertas dan uang koin.

UCAPAN TERIMA KASIH

Puji syukur penulis panjatkan kehadiran Tuhan Yang Maha Esa atas limpahan rahmat dan karunia oleh-Nya, sehingga makalah ini dapat diselesaikan dengan baik.

Pada bagian akhir dari makalah ini, penulis ingin menyampaikan ucapan terima kasih setulus-tulusnya kepada dosen pengampu mata kuliah Strategi Algoritma yaitu Imam Ekowicaksono, M. Si., dan Winda Yulita, M.Cs. yang telah memberi ilmu yang mendasari pembuatan makalah ini serta dukungan ketika proses pembuatan karya tulis ini.

Selain itu, penulis ingin menyampaikan rasa terima kasih kepada kedua orang tua penulis atas dukungan, doa, dan nasehat yang tak henti-hentinya diberikan kepada penulis sehingga makalah ini dapat diselesaikan dengan baik menggunakan kobaran semangat dan kekuatan yang disalurkan oleh pemberian dari kedua orang tua penulis.

Penulis sepenuhnya sadar bahwa hasil karya tulis berupa makalah ini masih jauh dari standar kesempurnaan. Maka dari itu, penulis ingin menyampaikan permohonan maaf atas kesalahan yang ada pada hasil maupun proses dari penulisan makalah ini. Penulis sangat menghargai pemberian kritik dan saran yang membangun untuk penyempurnaan proses penulisan karya ilmiah serupa yang dihadapi penulis di kemudian hari. Besar harapan penulis, semoga makalah yang telah ditulis ini dapat mendatangkan manfaat serta nilai yang positif bagi pihak pembaca yang membutuhkan.

REFERENCES

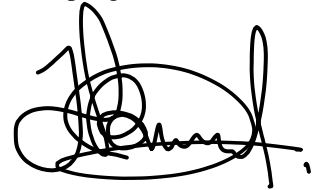
- [1] Massachusetts Institute of Technology, "Vending Machine", <http://web.mit.edu/2.744/studentSubmissions/humanUseAnalysis/keval/whatisvm.html>. Diakses pada 28 Maret 2022.
- [2] GeeksforGeeks, "Greedy Algorithms (General Structure and Applications)", <https://www.geeksforgeeks.org/greedy-algorithm-to-find-minimum-number-of-coins/>. Diakses pada 28 Maret 2022.

- [3] Paul Hype Page 7 Co, "Starting a Convenience Store in Indonesia", <https://www.paulhypepage.co.id/starting-a-convenience-store-in-indonesia/>. Diakses pada 28 Maret 2022.
- [4] I Massachusetts Institute of Technology, "Self-Checkout Machine", <https://web.mit.edu/2.744/www/Project/Assignments/humanUse/lynette/2-About%20the%20machine.html>. Diakses pada 28 Maret 2022.
- [5] Programiz, "Learn C++ Programming", <https://www.programiz.com/cpp-programming>. Diakses pada 28 Maret 2022.
- [6] Guru99, "C++ Programming: What is C++ | Learn Basic Concepts of C++", <https://www.guru99.com/cpp-tutorial.html>. Diakses pada 28 Maret 2022.
- [7] Tutorialspoint, "How does #include <bits/stdc++.h> work in C++?", <https://www.tutorialspoint.com/how-does-hashinclude-bits-stdcplusplus-h-work-in-cplusplus>. Diakses 28 Maret 2022.
- [8] CPlusPlus, "Arrays", <https://www.cplusplus.com/doc/tutorial/arrays/>. Diakses pada 28 Maret 2022.
- [9] GeeksforGeeks, "Vector in C++ STL", <https://www.geeksforgeeks.org/vector-in-cpp-stl/>. Diakses pada 28 Maret 2022.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandar Lampung, 28 Maret 2022



Nabila Muthia Putri 120140023