Jukompak (Jurnal Komputasi dan Pengembangan Aplikasi)
Vol. 1, No. 4, Oktober 2025, pp. 11~21

Perancangan dan Implementasi Program Penjualan Produk Matcha Menggunakan Bahasa Pemrograman C++ pada Matchuchu Griya

Nabila Nur Azzahrani[1], Dista Rika Susanti[2]
[1,2]Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro, Kota Semarang

Artikel Info

Kata kunci:

sistem penjualan
C++
pemrograman berorientasi objek
pemrograman terstruktur
rekayasa perangkat lunak

ABSTRAK

Penelitian ini bertujuan untuk merancang dan mengimplementasikan sistem aplikasi penjualan Matchuchu Griya menggunakan bahasa pemrograman C++ sebagai media penerapan konsep-konsep dasar pemrograman dalam konteks bisnis nyata. Sistem dikembangkan dengan mengintegrasikan berbagai paradigma pemrograman, meliputi pemrograman prosedural, pemrograman berorientasi objek, dan pemrograman terstruktur, guna menghasilkan aplikasi yang fungsional, terorganisir, serta mudah dipelihara. Metodologi pengembangan yang digunakan meliputi tahapan analisis kebutuhan, perancangan sistem, implementasi, dan pengujian untuk memastikan kesesuaian sistem dengan tujuan yang telah ditetapkan. Hasil penelitian menunjukkan bahwa sistem mampu menampilkan daftar produk, menerima masukan pengguna, memproses transaksi penjualan, serta menghitung total pembayaran secara akurat. Penerapan modularitas melalui fungsi-fungsi terpisah dan penggunaan struktur data yang kohesif meningkatkan keterbacaan kode, kemudahan pengujian, dan potensi pengembangan lanjutan. Selain sebagai solusi penjualan sederhana, sistem ini juga berfungsi sebagai media pembelajaran yang merepresentasikan penerapan prinsip-prinsip rekayasa perangkat lunak dan praktik pengkodean yang baik. Kesimpulan dari penelitian ini menunjukkan bahwa bahasa C++ dapat digunakan secara efektif untuk membangun sistem penjualan berskala kecil dengan arsitektur yang fleksibel dan dapat dikembangkan. Penelitian ini diharapkan dapat menjadi referensi akademik serta dasar bagi pengembangan sistem penjualan yang lebih kompleks di masa mendatang.

Penulis Korespondensi :

Nabila Nur Azzahrani,
Program Studi Teknik Informatika
Fakultas Ilmu Komputer
Universitas Dian Nuswantoro, Kota Semarang, 50131
Email: 111202516165@mhs.dinus.ac.id

1. PENDAHULUAN

Perkembangan teknologi informasi pada era digital saat ini telah membawa perubahan yang signifikan dalam berbagai aspek kehidupan manusia, termasuk di bidang pendidikan dan dunia usaha [1]. Pemanfaatan teknologi komputer tidak lagi terbatas pada kebutuhan administratif, tetapi telah berkembang menjadi sarana utama dalam mendukung proses pengolahan data dan pengambilan keputusan [2]. Dalam konteks kegiatan bisnis, penggunaan sistem komputer berperan penting dalam meningkatkan efisiensi dan akurasi proses transaksi, seperti penjualan, pembelian, dan pencatatan keuangan [3]. Proses yang sebelumnya dilakukan secara manual dengan risiko kesalahan yang tinggi kini dapat diotomatisasi melalui perangkat lunak sehingga lebih cepat, terstruktur, dan mudah dikelola.

Pemrograman komputer merupakan salah satu kompetensi fundamental yang harus dikuasai dalam pengembangan sistem berbasis teknologi informasi [4]. Bahasa pemrograman C++ sebagai salah satu bahasa pemrograman tingkat menengah banyak digunakan dalam pembelajaran karena memiliki kemampuan untuk

mengelola logika program secara terstruktur serta mendukung konsep pemrograman prosedural dan terstruktur [5]. Melalui C++, pengguna dapat memahami cara kerja dasar sebuah program, mulai dari pengolahan *input*, pemrosesan data, hingga menghasilkan *output* yang sesuai dengan kebutuhan pengguna [6]. Oleh karena itu, penguasaan bahasa pemrograman ini menjadi landasan penting dalam memahami pengembangan perangkat lunak secara lebih luas.

Dalam lingkungan akademik, khususnya pada mata kuliah pemrograman, mahasiswa tidak hanya dituntut untuk memahami konsep secara teoritis, tetapi juga mampu mengimplementasikannya ke dalam bentuk aplikasi yang nyata [7]. Penerapan konsep pemrograman melalui studi kasus menjadi pendekatan yang efektif untuk mengintegrasikan berbagai materi yang telah dipelajari, seperti mekanisme *input* dan *output*, percabangan logika, perulangan, penggunaan struktur data, serta penerapan fungsi dan prosedur [8]. Melalui pengembangan aplikasi sederhana, mahasiswa dapat melatih kemampuan analisis, logika berpikir sistematis, serta keterampilan pemecahan masalah yang dibutuhkan dalam bidang teknologi informasi.

Berdasarkan kebutuhan tersebut, pada penelitian ini dikembangkan sebuah program penjualan produk matcha pada Matchuchu Griya dengan menggunakan bahasa pemrograman C++. Program yang dirancang berbasis menu ini bertujuan untuk mensimulasikan proses transaksi penjualan secara terstruktur, mulai dari pemilihan produk, penentuan jumlah pembelian, hingga perhitungan total harga. Studi kasus penjualan dipilih karena merepresentasikan permasalahan yang umum dijumpai dalam kegiatan bisnis sehari-hari dan relevan untuk menguji penerapan konsep dasar pemrograman dalam sebuah sistem yang fungsional. Pengembangan program ini tidak hanya berfokus pada hasil akhir berupa aplikasi yang dapat dijalankan, tetapi juga pada proses perancangan dan implementasi yang sistematis. Setiap tahapan pengembangan program disusun dengan memperhatikan prinsip pemrograman terstruktur agar kode program mudah dipahami, dikembangkan, dan dipelihara. Selain itu, dokumentasi tertulis disusun sebagai bagian integral dari penelitian ini untuk menjelaskan alur perancangan program, logika yang digunakan, serta hasil implementasi secara komprehensif. Dokumentasi tersebut berfungsi sebagai media untuk menyampaikan proses berpikir dan metode yang digunakan dalam pengembangan aplikasi. Melalui penulisan artikel ilmiah ini, diharapkan dapat diperoleh gambaran yang jelas mengenai bagaimana konsep-konsep dasar pemrograman C++ diterapkan dalam pengembangan program penjualan berbasis menu. Selain itu, artikel ini juga diharapkan dapat menjadi referensi bagi pembaca dalam memahami proses perancangan aplikasi sederhana berbasis bahasa pemrograman C++ serta memberikan kontribusi dalam pengembangan pembelajaran pemrograman yang berorientasi pada penerapan praktis dan sistematis.

2. METODE

Bab ini menyajikan fondasi teoretis yang melandasi pengembangan sistem aplikasi yang dibahas dalam penelitian ini. Pembahasan mencakup konsep-konsep fundamental dalam bahasa pemrograman C++, termasuk paradigma pemrograman, struktur data, dan mekanisme kontrol program yang diimplementasikan dalam sistem.

2.1 Bahasa Pemrograman C++

C++ merupakan bahasa pemrograman tingkat menengah yang dikembangkan oleh Bjarne Stroustrup di Bell Laboratories pada tahun 1979 sebagai perluasan dari bahasa C dengan menambahkan fitur pemrograman berorientasi objek [9]. Bahasa ini awalnya diberi nama "*C with Classes*" sebelum akhirnya dinamai C++ pada tahun 1983, dengan simbol "++" yang merepresentasikan operator *increment* dalam bahasa C, mengindikasikan bahwa C++ adalah versi yang ditingkatkan dari C [10].

Gambar 1 Logo C++

C++ termasuk dalam kategori bahasa pemrograman multi-paradigma yang mendukung tiga paradigma utama [10]: (1) Pemrograman Prosedural. Paradigma ini mewarisi struktur dari bahasa C, di mana program disusun sebagai sekumpulan prosedur atau fungsi yang beroperasi pada data. Pendekatan ini menekankan pada urutan instruksi yang dieksekusi secara sekuensial, memungkinkan pemrogram untuk mengorganisir kode dalam unit-unit fungsional yang dapat digunakan kembali. (2) Pemrograman Berorientasi Objek (*Object-Oriented Programming*/OOP). Paradigma ini memungkinkan enkapsulasi data dan fungsi dalam entitas yang disebut objek, mendukung konsep-konsep seperti *inheritance* (pewarisan), *polymorphism* (polimorfisme), dan *encapsulation* (enkapsulasi). OOP memberikan pendekatan yang lebih natural dalam memodelkan masalah dunia nyata ke dalam struktur program. (3) *Generic Programming*. Melalui mekanisme *template*, C++ mendukung pemrograman generik yang memungkinkan penulisan kode yang dapat bekerja dengan berbagai tipe data tanpa duplikasi kode, meningkatkan *reusability* dan *type safety*.

C++ memiliki beberapa keunggulan signifikan yang menjadikannya pilihan utama untuk pengembangan aplikasi berkinerja tinggi. Pertama, bahasa ini menyediakan kontrol tingkat rendah terhadap memori dan perangkat keras, memungkinkan optimasi performa yang sangat baik [11]. Kedua, C++ mengombinasikan efisiensi eksekusi dengan abstraksi tingkat tinggi, memungkinkan pengembang untuk menulis kode yang efisien namun tetap bisa terawat. Aplikasi C++ mencakup berbagai domain, termasuk sistem operasi, *game engines*, aplikasi grafis, sistem *embedded*, *financial trading systems*, dan aplikasi *scientific computing* [12]. Performa tinggi dan fleksibilitas struktur C++ menjadikannya bahasa yang ideal untuk aplikasi-aplikasi yang membutuhkan efisiensi komputasi maksimal dan penggunaan sumber daya yang optimal.

2.2 Fungsi dalam Pemrograman

Fungsi merupakan unit dasar modularitas dalam pemrograman yang mengenkapsulasi sekumpulan instruksi untuk menjalankan tugas spesifik [13]. Dalam konteks *software engineering*, fungsi memfasilitasi prinsip "*separation of concerns*" dengan memisahkan logika program menjadi komponen-komponen independen yang dapat dikembangkan, diuji, dan dipelihara secara terpisah.

Implementasi fungsi dalam pengembangan perangkat lunak memberikan beberapa keuntungan fundamental: (1) Modularitas dan Struktur Program. Fungsi memungkinkan dekomposisi masalah kompleks menjadi sub-masalah yang lebih sederhana dan *manageable*, sesuai dengan prinsip *divide-and-conquer* dalam desain algoritma. Pendekatan modular ini meningkatkan organisasi kode dan memfasilitasi pemahaman struktur program secara keseluruhan; (2) *Reusability* dan *Code Efficiency*. Fungsi yang dirancang dengan baik dapat digunakan kembali di berbagai bagian program atau bahkan dalam proyek yang berbeda, mengurangi redundansi kode dan meningkatkan produktivitas pengembangan. Prinsip DRY (*Don't Repeat Yourself*) dapat tercapai melalui penggunaan fungsi yang efektif; (3) *Maintainability* dan *Debugging*. Dengan mengisolasi fungsionalitas spesifik dalam fungsi terpisah, proses identifikasi dan perbaikan *bug* menjadi lebih sistematis. Modifikasi pada satu fungsi tidak mempengaruhi bagian lain dari program selama antarmuka tetap konsisten.

Dalam konteks sistem yang dikembangkan, fungsi diimplementasikan untuk menangani berbagai aspek operasional, termasuk tampilan antarmuka pengguna (*display header* dan *footer*), perhitungan bisnis logika (komputasi total dan diskon), dan validasi data.

Tabel 1 Potongan Kode Fungsi

main.cpp

```
void header() {
    time_t now = time(0);
    tm *ltm = localtime(&now);

    int hari  = ltm->tm_mday;
    int bulan = ltm->tm_mon + 1;
    int tahun = ltm->tm_year + 1900;

    cout << "=====================================\n";
    cout << "       MATCHUCHU GRIYA\n";
    cout << "     Jl. Matcha No.1975, Brooklyn\n";
    cout << "Tanggal : " << hari << " ";

    string daftarBulan[] = {
        "Januari", "Februari", "Maret", "April", "Mei", "Juni",
        "Juli", "Agustus", "September", "Oktober", "November", "Desember"
```

```
    };

    cout << daftarBulan[bulan - 1] << " " << tahun << "\n";
    cout << "=====================================\n";
}

int hitungTotal(int jumlah, int harga) {
    return jumlah * harga;
}

float hitungDiskon(int total) {
    return (total > 200000) ? total * 0.20f : 0;
}

void footer() {
    cout << "=====================================\n";
    cout << "Terima kasih telah berbelanja di MATCHUCHU GRIYA!\n";
    cout << "PasswordWifi : belimatchadulu\n";
    cout << "=====================================\n";
}
```

Pendekatan fungsional ini memastikan setiap komponen memiliki tanggung jawab yang jelas dan terdefinisi dengan baik, sejalan dengan prinsip *Single Responsibility Principle* dalam SOLID *design principles* [14].

2.3 Struktur Data: *Struct*

Struct (*structure*) adalah tipe data bentukan yang memungkinkan pengelompokan beberapa variabel dengan tipe data yang berbeda atau sama dalam satu kesatuan logis [15]. Berbeda dengan *array* yang hanya dapat menyimpan elemen dengan tipe data homogen, *struct* menyediakan mekanisme untuk membuat *aggregate data type*s yang heterogen, memfasilitasi representasi entitas yang lebih kompleks dalam program.

Penggunaan *struct* memberikan beberapa keuntungan signifikan dalam organisasi data: (1) *Logical Grouping*. *Struct* memungkinkan pengelompokan data yang secara konseptual terkait dalam satu unit, meningkatkan kohesi data dan merefleksikan relasi antar data dengan lebih natural. Pendekatan ini sejalan dengan prinsip *information hiding* dan *data abstraction*; (2) *Code Clarity* dan *Expressiveness*. Dengan memberikan nama pada koleksi data terkait, *struct* meningkatkan *readability* kode dan *self-documentation*. Penggunaan *struct* membuat tujuan programmer lebih eksplisit dibandingkan dengan penggunaan variabel-variabel terpisah; (3) Efisiensi *Passing Parameter*. *Struct* memfasilitasi beberapa nilai-nilai terkait sebagai *single parameter* ke fungsi, mengurangi kompleksitas fungsi-fungsi dan meningkatkan pemeliharaan.

Dalam implementasi sistem yang dikembangkan, *struct* digunakan untuk merepresentasikan entitas data pembelian dengan mengenkapsulasi atribut-atribut seperti nama menu, harga per unit, dan jumlah item yang dibeli.

Tabel 2 Potongan Kode *Struct*

main.cpp

```
typedef struct {
    string namaMenu;
    int harga;
    int jumlah;
} Pembelian;
```

Struktur data ini memfasilitasi pengelolaan data transaksional dengan cara yang terorganisir, memungkinkan operasi-operasi seperti perhitungan subtotal, agregasi data, dan generasi laporan dilakukan dengan lebih efisien dan sistematis.

2.4 *Class* dan *Object*: Pemrograman Berorientasi Objek

Class merupakan *blueprint* atau *template* yang mendefinisikan karakteristik dan perilaku dari objek [16]. Dalam paradigma OOP, *class* mengenkapsulasi data (*attributes*/*member variables*) dan fungsi

(*methods*/*member functions*) yang beroperasi pada data tersebut. *Object* adalah instansiasi konkret dari *class*, merepresentasikan entitas spesifik dengan *state* dan *behavior* yang didefinisikan oleh *class* [16].

Implementasi *class* dalam C++ mengimplementasikan empat pilar fundamental OOP: (1) *Encapsulation* (Enkapsulasi). Class menyediakan mekanisme untuk menyembunyikan detail implementasi internal dan hanya mengekspos *interface* yang diperlukan melalui *access specifiers* (*public*, *private*, *protected*). Prinsip ini mendukung penyembunyian informasi dan mengurangi *coupling* antar komponen; (2) *Abstraction* (Abstraksi). *Class* memungkinkan pemodelan konsep-konsep abstrak dari domain masalah ke dalam representasi programatik, fokus pada karakteristik esensial sambil menyembunyikan kompleksitas yang tidak relevan; (3) *Inheritance* (Pewarisan). Meskipun tidak diimplementasikan dalam sistem yang dibahas, C++ mendukung inheritance yang memungkinkan *class* baru mewarisi properti dan *method* dari *class* yang ada, memfasilitasi *code reuse* dan *hierarchical classification*; (4) *Polymorphism* (Polimorfisme). Kemampuan objek untuk mengambil berbagai bentuk, memungkinkan *interface* yang sama untuk diimplementasikan dengan cara yang berbeda pada *class*-*class* yang berbeda.

Dalam sistem yang dikembangkan, *class* Matcha digunakan untuk merepresentasikan entitas minuman matcha sebagai objek bisnis dengan atribut-atribut seperti jenis minuman dan harga.

Tabel 3 Potongan Kode *Class* dan *Object*

main.cpp

```
class Matcha {
private:
   string nama;
   int harga;
public:
   Matcha(string n, int h) : nama(n), harga(h) {}
   string getNama() { return nama; }
   int getHarga() { return harga; }
};
```

Pendekatan OOP ini memberikan beberapa keuntungan: pertama, memfasilitasi pemodelan domain bisnis yang lebih natural dan intuitif; kedua, memungkinkan ekstensi fungsionalitas di masa depan tanpa memodifikasi kode yang ada secara signifikan (*Open-Closed Principle*); ketiga, meningkatkan pemeliharaan melalui enkapsulasi yang jelas antara data dan operasi yang berkaitan dengan entitas matcha.

2.5 Struktur Kontrol: Percabangan dan Perulangan

Struktur kontrol merupakan konstruksi fundamental dalam pemrograman yang mengatur alur eksekusi program [17]. Berbeda dengan eksekusi sekuensial sederhana, struktur kontrol memungkinkan program untuk membuat keputusan (*decision making*) dan melakukan repetisi (*iteration*), memberikan fleksibilitas dan kemampuan komputasi yang kuat.

Tabel 4 Potongan Kode Percabangan dan Perulangan

main.cpp

```
for (int i = 0; i < 3; i++) {
      cout << i + 1 << ". " << matchas[i].getNama()
          << " (Rp" << matchas[i].getHarga() << ")\n";
   }

   cout << "\nBerapa banyak jenis matcha yang ingin Anda pilih? ";
   cin >> banyakPilihan;

   if (banyakPilihan < 1 || banyakPilihan > 3) {
      cout << "Maaf... jumlah pilihan hanya boleh 1-3.\n";
      return 0;
   }
for (int i = 0; i < banyakPilihan; i++) {

      int pilihan, jumlah;
```

```
        cout << "\nPilih rasa (1-3): ";
        cin >> pilihan;

        while (pilihan < 1 || pilihan > 3) {
            cout << "Maaf, pilihan tidak valid. Silakan pilih 1-3: ";
            cin >> pilihan;
        }

        cout << "Masukkan jumlah pembelian: ";
        cin >> jumlah;

        data[i].namaMenu = matchas[pilihan - 1].getNama();
        data[i].harga    = matchas[pilihan - 1].getHarga();
        data[i].jumlah   = jumlah;

        total += hitungTotal(jumlah, data[i].harga);
    }
for (int i = 0; i < banyakPilihan; i++) {

        int subtotal = hitungTotal(data[i].jumlah, data[i].harga);

        cout << i + 1 << "    "
             << data[i].namaMenu << "      "
             << "Rp" << data[i].harga << "      "
             << data[i].jumlah << "        "
             << "Rp" << subtotal << endl;
    }
if (diskon > 0)
        cout << "Diskon              : Rp" << diskon << endl;

    cout << "Total setelah diskon : Rp" << totalAkhir << endl;
```

Percabangan merupakan mekanisme pengambilan keputusan dalam program berdasarkan evaluasi kondisi *boolean* [18]. C++ menyediakan beberapa konstruksi percabangan:

1. *If-Else Statement*. Struktur kondisional dasar yang memungkinkan eksekusi blok kode berbeda berdasarkan nilai-nilai yang benar dari kondisi yang dievaluasi. Implementasi *if-else* mendukung *decision tree complexity* yang dapat digunakan untuk *logic* bisnis kompleks.
2. *Switch-Case Statement*. Alternatif untuk banyak *if-else* yang lebih efisien ketika membandingkan satu variabel terhadap beberapa nilai konstan, memberikan keuntungan dalam *readability* dan performa untuk skenario-skenario tertentu.
3. *Conditional Operator* (*Ternary*). Operator kondisional singkat yang memfasilitasi *assignment* kondisional dalam satu ekspresi, berguna untuk kondisi sederhana yang membutuhkan keringkasan.

Perulangan memungkinkan eksekusi berulang dari sekumpulan instruksi sampai kondisi terminasi terpenuhi [19]. Mekanisme ini esensial untuk memproses koleksi, perhitungan berulang, dan mengimplementasikan algoritme:

1. *For Loop*. Struktur iterasi yang ideal ketika jumlah iterasi diketahui sebelumnya, menyediakan *initialization*, *condition checking*, dan *increment*/*decrement* dalam satu *statement*.
2. *While* dan Do-While Loop. Konstruksi iterasi yang bergantung pada kondisi boolean, di mana *while* melakukan *pre-test* (kondisi diperiksa sebelum eksekusi) dan *do-while* melakukan *post-test* (kondisi diperiksa setelah eksekusi), memberikan fleksibilitas untuk berbagai skenario iterasi.

Dalam konteks sistem yang dikembangkan, struktur kontrol memiliki peran kritis: (1) Validasi *Input*. Percabangan digunakan untuk memverifikasi bahwa data *input* dari *user* memenuhi *constraint* yang ditentukan (*range validation*, *type checking*, *business rule validation*), memastikan data integrity dan preventing error propagation; (2) Pengolahan Data Pembelian. Perulangan difungsikan untuk iterasi melalui koleksi data pembelian, melakukan agregasi (penjumlahan subtotal), transformasi (menerapkan diskon), dan pembuatan

output (menampilkan daftar rincian). Kombinasi percabangan dan perulangan memungkinkan implementasi logika bisnis yang kompleks seperti diskon bersyarat, penetapan harga berjenjang, dan pemrosesan berbasis kategori; (3) *Error Handling* dan *Robustness*. Struktur kontrol juga digunakan untuk menerapkan mekanisme penanganan kesalahan, memastikan program dapat menangani kondisi tidak terduga dan menjaga stabilitas sistem.

2.6 Kontribusi Teoretis

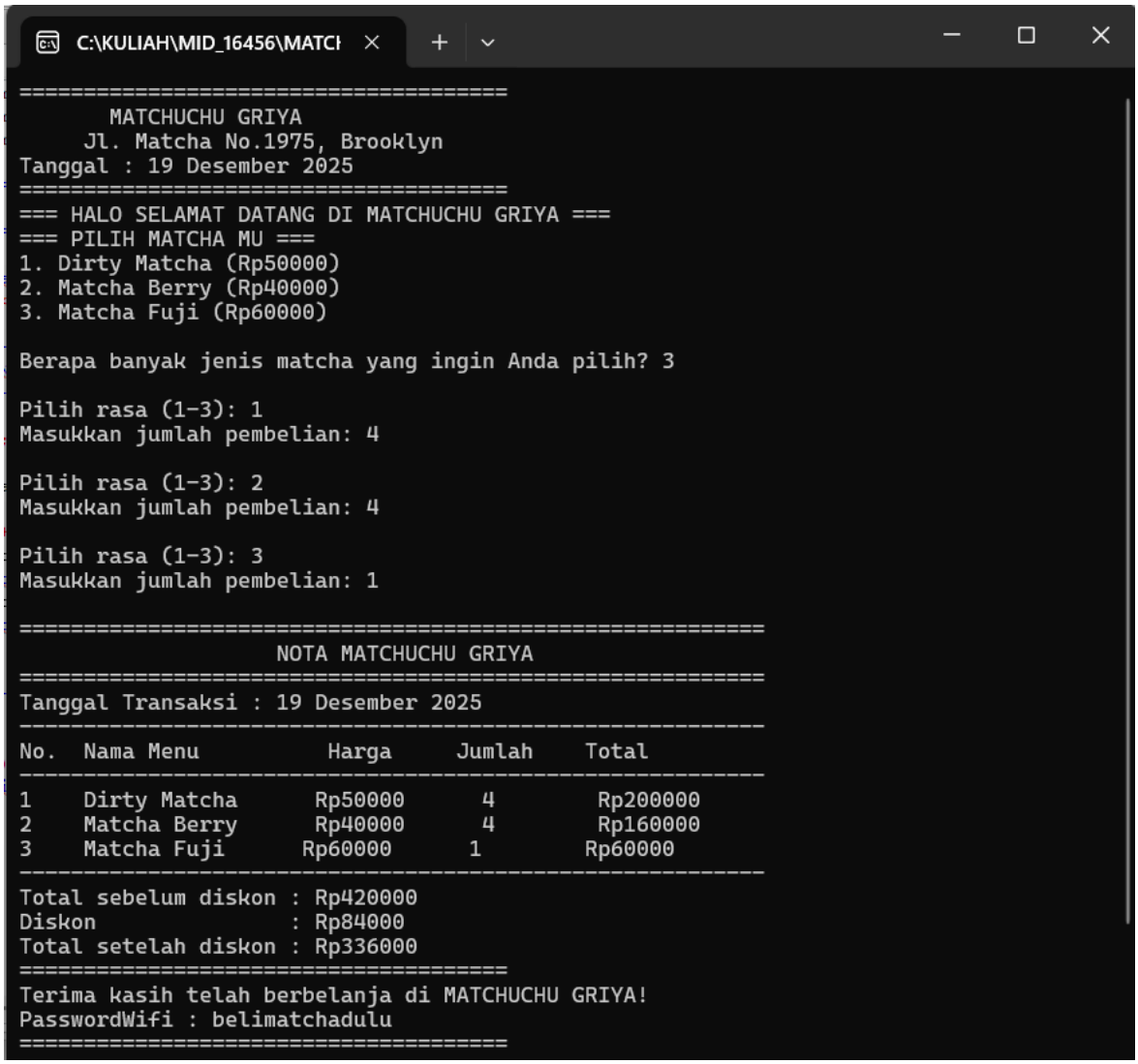
Landasan teori yang dipaparkan dalam bab ini memberikan fondasi konseptual yang komprehensif untuk memahami desain dan implementasi sistem. Integrasi berbagai konsep pemrograman, dari level rendah seperti *struct* hingga abstraksi tingkat tinggi seperti *class* yang mendemonstrasikan pendekatan *multi-layer* dalam *software development* yang seimbang antara efisiensi, kemudahan pemeliharaan, dan skalabilitas. Kontribusi utama dari kerangka teoretis ini adalah penyediaan justifikasi ilmiah untuk pilihan desain dalam implementasi sistem, menunjukkan bagaimana prinsip-prinsip *software engineering* fundamental dapat diterapkan dalam konteks aplikasi bisnis praktis. Pemahaman mendalam tentang konstruksi bahasa pemrograman ini tidak hanya memfasilitasi pengembangan sistem yang kuat dan efisien, tetapi juga memberikan fondasi untuk ekstensi dan peningkatan di masa depan sesuai dengan evolusi kebutuhan bisnis.

3. PEMBAHASAN HASIL

Bab ini menyajikan pembahasan secara komprehensif mengenai hasil perancangan dan implementasi program penjualan Matchuchu Griya yang telah dijelaskan pada bab sebelumnya. Fokus utama pembahasan diarahkan pada bagaimana rancangan sistem diterapkan ke dalam bentuk program yang dapat dijalankan, serta bagaimana setiap komponen program bekerja dalam mendukung proses transaksi penjualan secara keseluruhan. Pembahasan dimulai dengan penjelasan umum mengenai karakteristik dan tujuan program untuk memberikan gambaran awal kepada pembaca terkait sistem yang dikembangkan. Selanjutnya, struktur dan arsitektur program diuraikan guna menunjukkan penerapan konsep pemrograman C++ yang digunakan, diikuti dengan penjabaran alur eksekusi program sebagai representasi proses transaksi yang terjadi. Bab ini juga membahas implementasi perhitungan transaksi beserta analisis hasil yang diperoleh dari pengujian program. Pada bagian akhir, disajikan evaluasi terhadap kinerja program melalui identifikasi kelebihan dan keterbatasan sebagai dasar untuk pengembangan lebih lanjut. Dengan struktur tersebut, Bab 3 diharapkan mampu memberikan pemahaman yang sistematis dan mendalam mengenai hasil implementasi program, sekaligus menunjukkan keterkaitan antara perancangan, pelaksanaan, dan capaian yang dihasilkan.

3.1 Deskripsi Umum Program

Program penjualan Matchuchu Griya merupakan aplikasi berbasis *console* yang dirancang untuk mensimulasikan proses transaksi penjualan minuman matcha secara terkomputerisasi. Program ini dikembangkan menggunakan bahasa pemrograman C++ dengan tujuan utama untuk mengimplementasikan konsep-konsep dasar pemrograman terstruktur dan berorientasi objek dalam konteks permasalahan nyata, khususnya pada sistem kasir sederhana.



Gambar 2 Hasil Implementasi Program

Pada Gambar 2 tersebut menampilkan hasil eksekusi program sistem pemesanan minuman "Matchuchu Griya" berbasis C++ melalui tampilan command prompt. Pada bagian awal, sistem menampilkan identitas usaha yang meliputi nama toko, alamat, serta tanggal transaksi. Informasi ini berfungsi sebagai header atau identitas transaksi yang memberikan konteks administratif terhadap proses pembelian yang dilakukan. Selanjutnya, program menampilkan menu pilihan minuman matcha yang tersedia beserta harga masing-masing produk, yaitu Dirty Matcha seharga Rp50.000, Matcha Berry seharga Rp40.000, dan Matcha Fuji seharga Rp60.000. Sistem kemudian meminta pengguna untuk memasukkan jumlah jenis menu yang ingin dipilih. Pada contoh eksekusi tersebut, pengguna memilih tiga jenis menu. Untuk setiap pilihan menu, sistem meminta input nomor menu (1–3) serta jumlah pembelian. Proses ini menunjukkan implementasi struktur perulangan (looping) yang memungkinkan pengguna melakukan pemesanan lebih dari satu jenis produk dalam satu transaksi. Setelah seluruh data pembelian dimasukkan, program secara otomatis menampilkan nota transaksi yang memuat rincian pembelian dalam bentuk tabel. Tabel tersebut terdiri atas nomor urut, nama menu, harga satuan, jumlah pembelian, dan total harga per item. Perhitungan total dilakukan dengan mengalikan harga satuan dengan jumlah pembelian pada masing-masing menu. Berdasarkan data yang dimasukkan, total pembelian sebelum diskon adalah Rp420.000. Sistem kemudian menerapkan mekanisme diskon secara otomatis, di mana besaran diskon yang diberikan adalah Rp84.000. Nilai tersebut dihitung sebagai persentase tertentu dari total belanja, sehingga menghasilkan total pembayaran akhir sebesar Rp336.000 setelah diskon. Proses ini menunjukkan penerapan operasi aritmatika serta logika percabangan dalam program untuk

menentukan besaran potongan harga. Sistem kemudian menerapkan mekanisme diskon secara otomatis, di mana besaran diskon yang diberikan adalah Rp84.000. Nilai tersebut dihitung sebagai persentase tertentu dari total belanja, sehingga menghasilkan total pembayaran akhir sebesar Rp336.000 setelah diskon. Proses ini menunjukkan penerapan operasi aritmatika serta logika percabangan dalam program untuk menentukan besaran potongan harga.

3.2 Struktur dan Arsitektur Program

Struktur program dirancang secara modular untuk meningkatkan keterbacaan kode (*readability*), kemudahan pemeliharaan (*maintainability*), serta potensi pengembangan lanjutan. Secara umum, struktur utama program terdiri atas beberapa komponen berikut: (1) Fungsi *header* dan *footer*, yang berfungsi untuk menampilkan identitas toko dan penutup nota transaksi secara konsisten, sehingga meningkatkan kerapian tampilan *output*. (2) Fungsi perhitungan total dan diskon, yang mengenkapsulasi logika perhitungan transaksi agar tidak tercampur dengan proses *input* dan *output*. (3) *Struct* data pembelian, yang digunakan untuk menyimpan data transaksi seperti nama menu, harga satuan, jumlah pembelian, dan subtotal, sehingga pengelolaan data menjadi lebih terorganisir. (4) *Class* daftar menu matcha, yang merepresentasikan penerapan konsep pemrograman berorientasi objek dalam pengelolaan data menu, sekaligus mempermudah penambahan atau perubahan daftar produk. (5) Fungsi utama (main), yang berperan sebagai pengendali alur program secara keseluruhan, mulai dari pemanggilan fungsi hingga interaksi dengan pengguna. Pendekatan ini sejalan dengan prinsip *modular programming* yang menekankan pemisahan tanggung jawab setiap bagian program agar lebih sistematis dan mudah dipahami.

3.3 Alur Eksekusi Program

Alur kerja program disusun secara berurutan dan logis untuk merepresentasikan proses transaksi penjualan yang umum terjadi pada sistem kasir. Tahapan alur program dapat dijelaskan sebagai berikut:

1. Program diawali dengan menampilkan *header* yang memuat identitas toko Matchuchu Griya.
2. Program kemudian menampilkan daftar menu matcha beserta harga masing-masing.
3. Pengguna diminta untuk menentukan jumlah jenis menu matcha yang akan dibeli.
4. Untuk setiap jenis menu, pengguna memilih menu yang diinginkan dan memasukkan jumlah pembelian.
5. Program melakukan perhitungan subtotal, total belanja, serta diskon berdasarkan ketentuan yang berlaku.
6. Hasil perhitungan ditampilkan dalam bentuk nota transaksi yang terstruktur dan informatif.
7. Program menutup proses dengan menampilkan *footer* sebagai penanda akhir transaksi.

Alur tersebut dirancang agar mudah dipahami oleh pengguna sekaligus mencerminkan proses transaksi nyata, sehingga meningkatkan relevansi program terhadap studi kasus yang diangkat.

3.4 Implementasi dan Analisis Perhitungan Transaksi

Implementasi perhitungan transaksi dilakukan melalui pendekatan matematis sederhana namun efektif. Total belanja dihitung dengan mengalikan harga satuan setiap menu dengan jumlah pembelian, kemudian menjumlahkan seluruh subtotal. Selanjutnya, sistem menerapkan kebijakan diskon sebesar 20% apabila total belanja melebihi Rp200.000. Secara algoritmik, mekanisme ini memastikan bahwa proses perhitungan berlangsung secara akurat dan konsisten. Total akhir yang harus dibayarkan oleh pengguna diperoleh dari pengurangan total belanja dengan nilai diskon yang diberikan. Penerapan logika kondisional (*conditional statement*) pada proses diskon menunjukkan pemanfaatan struktur kontrol dalam C++ untuk menangani aturan bisnis tertentu.

3.5 Analisis Kinerja dan Hasil Program

Berdasarkan hasil pengujian yang dilakukan, program telah berfungsi sesuai dengan tujuan perancangan. Validasi *input* yang diterapkan mampu mencegah kesalahan pemilihan menu, sehingga meminimalkan potensi kesalahan perhitungan. Selain itu, pembagian fungsi dan penggunaan *class* memberikan struktur kode yang lebih rapi, terorganisir, dan mudah dipahami, baik oleh pengembang maupun oleh pihak lain yang akan mempelajari kode tersebut. Dari sisi hasil, program berhasil menampilkan nota transaksi yang memuat informasi lengkap, seperti tanggal transaksi, daftar menu yang dibeli, harga satuan, jumlah pembelian, total belanja, diskon, dan total pembayaran akhir. Hal ini menunjukkan bahwa program mampu memenuhi kebutuhan dasar sistem penjualan sederhana.

3.6 Kelebihan dan Keterbatasan Program

Program yang dikembangkan memiliki beberapa kelebihan utama, antara lain struktur kode yang terorganisir dengan baik, kemudahan dalam pengembangan lanjutan seperti penambahan menu, serta penerapan konsep dasar pemrograman C++ secara komprehensif, mencakup *struct*, *class*, fungsi, dan struktur kontrol. Namun demikian, program ini juga memiliki beberapa keterbatasan. Program masih berjalan dalam lingkungan *console*, sehingga tampilan antarmuka masih bersifat teks dan kurang interaktif. Selain itu, sistem belum mendukung penyimpanan data transaksi ke dalam *file* atau basis data, serta belum menyediakan fitur pembatalan pesanan yang dapat meningkatkan fleksibilitas penggunaan.

3.7 Kontribusi terhadap Bidang Pembelajaran Pemrograman

Secara keseluruhan, program penjualan Matchuchu Griya memberikan kontribusi nyata sebagai studi kasus pembelajaran pemrograman C++. Program ini menunjukkan bagaimana konsep-konsep dasar pemrograman dapat diterapkan untuk menyelesaikan permasalahan nyata di bidang sistem informasi penjualan. Dengan demikian, program ini dapat dijadikan sebagai referensi atau dasar pengembangan lebih lanjut, baik untuk keperluan akademik maupun sebagai prototipe awal sistem kasir yang lebih kompleks.

4. KESIMPULAN

Bab penutup ini merangkum keseluruhan pembahasan penelitian dengan mengintegrasikan temuan utama dari proses pengembangan sistem serta memberikan rekomendasi untuk pengembangan selanjutnya. Bagian ini menekankan pencapaian penelitian, kontribusinya terhadap pembelajaran dan dokumentasi akademik, validasi metodologi yang digunakan, serta arah peningkatan sistem secara berkelanjutan. Penelitian ini berhasil mengimplementasikan sistem aplikasi penjualan Matchuchu Griya menggunakan bahasa pemrograman C++ dengan pendekatan multi-paradigma, meliputi pemrograman prosedural, berorientasi objek, dan terstruktur. Integrasi paradigma tersebut menunjukkan fleksibilitas C++ serta kemampuan penerapan prinsip rekayasa perangkat lunak secara praktis. Penerapan modularitas melalui fungsi-fungsi yang jelas menghasilkan struktur program yang terorganisir, mudah dibaca, dan mudah dipelihara. Dari aspek pengelolaan data dan logika program, penggunaan struktur data meningkatkan kohesi dan kemudahan manipulasi informasi transaksi. Pemodelan domain bisnis melalui kelas berorientasi objek memberikan abstraksi yang intuitif serta mendukung prinsip enkapsulasi dan pengurangan ketergantungan antarkomponen. Struktur kontrol alur program yang diterapkan memungkinkan validasi masukan, pengambilan keputusan yang tepat, serta pemrosesan transaksi yang efisien.

Sistem ini juga memberikan kontribusi penting dalam konteks pembelajaran dan dokumentasi akademik. Aplikasi yang dikembangkan dapat dimanfaatkan sebagai media pembelajaran untuk memahami penerapan konsep pemrograman dalam skenario bisnis nyata, sekaligus mendokumentasikan praktik terbaik pengembangan perangkat lunak. Keberhasilan implementasi ini memvalidasi metodologi pengembangan yang sistematis dan terstruktur, yang terbukti efektif dalam menghasilkan sistem yang fungsional dan dapat dikembangkan. Berdasarkan hasil evaluasi, disarankan pengembangan lanjutan berupa penerapan penyimpanan data permanen, peningkatan antarmuka pengguna, serta perluasan fitur pembayaran dan keamanan sistem. Selain itu, adopsi pola desain, praktik pengembangan tangkas, serta strategi pengujian dan optimasi kinerja dapat meningkatkan kualitas sistem. Secara keseluruhan, penelitian ini berhasil mencapai tujuannya dan menyediakan fondasi yang kuat bagi pengembangan sistem penjualan yang lebih andal, skalabel, dan relevan dengan kebutuhan bisnis di masa depan.

REFERENCES

[1] D. E. Syafitri and F. A.-I. A. Putra, “Artikel Jurnal Perancangan UI/UX Mobile App Interaktif untuk Meningkatkan Promosi dan Penjualan Produk F&B pada Eatery Cafe,” *SMATIKA J.*, vol. 14, no. 02, pp. 250–260, Dec. 2024, doi: 10.32664/smatika.v14i02.1318.

[2] M. R. Sihombing, A. I. Permana, and H. Herdianto, “Rancang Bangun Aplikasi Data Penjualan Pada Ausy Coffee Berbasis Web,” *J. Minfo Polgan*, vol. 14, no. 1, pp. 427–434, Apr. 2025, doi: 10.33395/jmp.v14i1.14744.

[3] Kevin Gustian Yulius *et al.*, “Pelatihan Penggunaan Teknologi Informasi dalam Pemasaran Makanan dan Minuman di Kampung Tehyan Tangerang,” *J. SOLMA*, vol. 12, no. 2, pp. 516–521, Aug. 2023, doi: 10.22236/solma.v12i2.11984.

[4] P. Primawati, F. Rozi, R. E. Wulansari, F. Prasetya, A. Ambiyar, and F. Efendi, “EFEKTIVITAS PENERAPAN FLIPPED CLASSROOM DENGAN IMAGE PROCESSING UNTUK MENINGKATKAN KETERAMPILAN PEMROGRAMAN KOMPUTER PADA MAHASISWA TEKNIK MESIN,” *J. Vokasi Mek. VoMek*, vol. 6, no. 3, pp. 286–291, Aug. 2024, doi: 10.24036/vomek.v6i3.732.

[5] Z. Chen, Y. Zhu, and Z. Wang, “Design and Implementation of an Aspect-Oriented C Programming Language,” *Proc. ACM Program. Lang.*, vol. 8, no. OOPSLA1, pp. 642–669, Apr. 2024, doi: 10.1145/3649834.

[6] T. M. M. -, "C Programming: Structured Programming Language," *Int. J. Multidiscip. Res.*, vol. 6, no. 6, p. 30657, Nov. 2024, doi: 10.36948/ijfmr.2024.v06i06.30657.
[7] B. Pramono, "IMPLEMENTASI ALGORITMA FISHER-YATES SHUFFLE PADA APLIKASI QUIZ ONLINE MATERI PEMROGRAMAN DASAR," *AnoaTIK J. Teknol. Inf. Dan Komput.*, vol. 2, no. 1, pp. 22–29, Jun. 2024, doi: 10.33772/anoatik.v2i1.31.
[8] J. Bata, "Pengembangan Pembelajaran Daring Menggunakan Metode ADDIE Pada Topik Computational Thinking dan Pemrograman Dasar," *JIIP - J. Ilm. Ilmu Pendidik.*, vol. 7, no. 5, pp. 4691–4696, May 2024, doi: 10.54371/jiip.v7i5.4421.
[9] B. A. Saputra, S. C. Candra, F. B. Wijaya, K. M. Suryaningrum, and H. A. Saputri, "Comparative Analysis of Binary and Interpolation Search Algorithms on Integer Data Using C Programming Language," *2023 Int. Conf. Inf. Manag. Technol. ICIMTech*, pp. 340–345, Aug. 2023, doi: 10.1109/ICIMTech59029.2023.10277955.
[10] A. Alagarsamy, "C Programming Language Better Understanding," *Int. J. Multidiscip. Res.*, vol. 7, no. 3, p. 49293, Jun. 2025, doi: 10.36948/ijfmr.2025.v07i03.49293.
[11] A.R.Rajabov, "ONE-DIMENSIONAL ARRAYS IN THE C++ PROGRAMMING LANGUAGE," *Introd. New Innov. Technol. Educ. Pedagogy Psychol.*, vol. 2, no. 5, pp. 90–97, May 2025.
[12] A. Nazri, H. Liyana, I. Hasanah, and Z. Ibrahim, "Tales of C++ Worlds: C++ Programming Language Game-Based Learning," *Multidiscip. Appl. Res. Innov.*, vol. 5, no. 1, pp. 223–231, Jan. 2024.
[13] V. V. Ponggawa, U. B. Santoso, G. A. Talib, M. A. Lamia, A. R. A. Manuputty, and M. F. Yusuf, "Comparative Study of C++ and C# Programming Languages," *J. Syntax Admiration*, vol. 5, no. 12, pp. 5743–5748, Dec. 2024, doi: 10.46799/jsa.v5i12.1926.
[14] B. Stroustrup, *Programming: Principles and Practice Using C++*. Addison-Wesley Professional, 2024.
[15] B. Stroustrup, *A Tour of C++*. Addison-Wesley Professional, 2022.
[16] B. Stroustrup, *Programming: Principles and Practice Using C++*. Addison-Wesley Professional, 2024.
[17] S. bin Uzayr, Ed., *Mastering C++ Programming Language: A Beginner's Guide*. Boca Raton: CRC Press, 2022. doi: 10.1201/9781003214762.
[18] N. V. Do and T. T. Mai, "A Knowledge Representation Model for Designing the Knowledge Querying System in Programming Language C/C++," in *2023 RIVF International Conference on Computing and Communication Technologies (RIVF)*, Dec. 2023, pp. 366–371. doi: 10.1109/RIVF60135.2023.10471842.
[19] K. A. Onyango and G. M. Wambugu, "Comparative Analysis on the Evaluation of the Complexity of C, C++, Java, PHP and Python Programming Languages based on Halstead Software Science," Mar. 2023, Accessed: Jan. 08, 2026. [Online]. Available: http://repository.mut.ac.ke:8080/xmlui/handle/123456789/6419