RESPONSI PRAKTIKUM STRUKTUR DATA DAN ALGORITMA 2023

- Identitas

  Nama : Muhammad Ryan Fikri Fakhrezi
  NIM : L0122114
  Kelas : C
  Judul program : Bookstore Simulator
  Deskripsi program : Program ini mensimulasikan kegiatan jual beli di toko buku

- Dokumentasi program

  Program ini merupakan suatu program simulasi toko buku, diawal program akan mendeklarasikan variabel role yang digunakan untuk menampung pilihan role yang dipilih user, disini juga dideklarasikan variabel people yang akan terus bertambah seiring bertambahnya customer. kemudian program akan melakukan parsing pada file xml dan mendata semua id yang terdapat pada file xml, dan menaruhnya pada set yang bernama all_book_id. setelah semua preparasi sudah dilakukan, program akan memasuki suatu infinity loop dan memberikan pilihan 2 role, apakah user ingin menjadi kasir ataukah customer

```
int main(){
    int role;
    int people = 1;
    XMLDataHandler forCollectId;
    forCollectId.collectAllBookIds();
    while(true){
        clear_screen();
        bool canICloseTheStore = false;
        std::cout << "| === WELCOME TO BOOKSTORE SIMULATOR === |\n";
        std::cout << "| CHOOSE YOUR ROLE                      |\n";
        std::cout << "| 1] Customer                           |\n";
        std::cout << "| 2] Cashier                            |\n";
        std::cout << "| What's your role : ";
```

  kemudian program akan meminta input dari user mengenai role yang diinginkan, program akan terus meminta input role selama user memasukkan pilihan yang tidak valid

```
        //check if the role is valid
        do{
            if (!(std::cin >> role))
                {
                    std::cout << "| Invalid input! Please enter an integer
only." << '\n';
                    std::cin.clear();
```

```
                    std::cin.ignore(std::numeric_limits<std::streamsize>::max(
), '\n');
                    continue;
                }

            if ((role >= 1) && (role <= 2))
                break;

            std::cout << "| Please enter a valid number (1 or 2)" <<
std::endl;
        }while (true);
```

Jika user memasukkan pilihan yang valid, program akan menjalankan switch statement berdasarkan role yang dipilih, jika user memasukkan pilihan 1, maka block kode pada case 1 akan dijalankan, disini program akan memberitahukan bahwa user sedang bertindak sebagai customer dan memanggil fungsi clear_screen() yang berguna untuk membersihkan terminal, dan fungsi waitEnter() supaya user semppat membaca pemberitahuan dan program baru akan menjalankan baris selanjutnya apabila user sudah menekan tombol enter.

```
        switch(role){
            case 1:
            {
                std::cout << "| Now, you'll act like a customer who wants to
buy books.\n";
                WaitEnter();
                clear_screen();
```

Setelah itu, program akan membuat objek baru dengan dynamic memory allocation dari class buyer, didalam class tersebut terdapat sebuah stack Bernama cart yang berguna untuk menampung data bertipe string yang merupakan id dari produk yang akan dipilih oleh customer nantinya. Untuk mengisi stack tersebut, program akan memanggil fungsi customer

```
                buyer *Buyer = new buyer();
                Buyer->cart = customer();
```

Didalam fungsi customer, fungsi ini akan mengembalikan data dalam bentuk stack. Disini dideklarasikan stack dengan nama cart yang berisi id buku yang dipilih oleh customer, dan graph berupa edge list yang dibuat oleh fungsi generateGraphForSimilarities

```
std::stack<std::string> customer(){
    XMLDataHandler accessXml;
    std::stack<std::string> cart;
    std::list<similar> recommendation =
accessXml.generateGraphForSimilarities();
```

fungsi generateGraphForSimilarities tampak seperti dibawah, disini program akan melakukan loop antar buku untuk mengecek apakah ada kesamaan dalam hal author maupun genre, fungsi ini mengembalikan data dalam bentuk edge list

```
    std::list<similar> generateGraphForSimilarities(){
        std::list<similar> similarity_list;
```

```
        for (rapidxml::xml_node<> * book_node_A = root_node->first_node("book");
             book_node_A;
             book_node_A = book_node_A->next_sibling())
        {
            for (rapidxml::xml_node<> * book_node_B = book_node_A->next_sibling();
                 book_node_B;
                 book_node_B = book_node_B->next_sibling())
            {
                std::string Book_A_author = book_node_A->first_node("author")->value();
                std::string Book_A_genre = book_node_A->first_node("genre")->value();
                std::string Book_B_author = book_node_B->first_node("author")->value();
                std::string Book_B_genre = book_node_B->first_node("genre")->value();
                if(Book_A_author == Book_B_author){
                    similarity_list.push_back({book_node_A->first_attribute("id")->value(), book_node_B->first_attribute("id")->value(), "author"});
                    similarity_list.push_back({book_node_B->first_attribute("id")->value(), book_node_A->first_attribute("id")->value(), "author"});
                }
                if(Book_A_genre == Book_B_genre){
                    similarity_list.push_back({book_node_A->first_attribute("id")->value(), book_node_B->first_attribute("id")->value(), "genre"});
                    similarity_list.push_back({book_node_B->first_attribute("id")->value(), book_node_A->first_attribute("id")->value(), "genre"});
                }
            }
        }
        return similarity_list;
    }
```

Kemudian program akan meminta keyword dari user mengenai buku yang ingin dicari menggunakan fungsi search, selama hasil search tidak ditemukan, program akan selalu meminta user untuk memasukkan keyword

```
        std::cout << "| ======================================== WELCOME TO FOOBAR BOOKSTORE ======================================== |" << std::endl;
        while(true){
            std::cout << "| enter the book's title/author/genre : ";
            getline(std::cin, find);
            found = accessXml.search(find);
```

```
            //not found
            if(found.size()!=0){break;}
            std::cout << "Nothing found\n";
        }
```

Fungsi search tampak seperti dibawah, disini program melakukan loop antar buku, dan setiap buku dilakukan loop antar metadata buku untuk mengecek adanya kesesuaian dengan input keyword user. Jika ditemukan kesesuaian, program akan menambahkan id buku tersebut kedalam list yang Bernama found. Setelah iterasi selesai, program akan mengembalikan data list found.

```
    std::list<std::string> search(std::string keyword){
        std::list<std::string> found;
        // Iterate over the book nodes
        for (rapidxml::xml_node<> * book_node = root_node->first_node("book");
                    book_node;
                    book_node = book_node->next_sibling())
        {
        // Iterate over the book data
            for(rapidxml::xml_node<> * Book_metadata = book_node-
>first_node();
                    Book_metadata;
                    Book_metadata = Book_metadata->next_sibling())
            {
                // Added the book's id to the search result whenever there's
matched result
                if(Book_metadata->value()==keyword){
                    found.push_back(book_node->first_attribute("id")-
>value());
                }
            }
        }
        return found;
```

Kembali ke fungsi customer, apabila ditemukan data buku yang cocok dengan keyword, dilakukan iterasi pada list dan mengeprint data buku di setiap iterasi dengan fungsi printBookDataBasedOfId

```
        //found
        while(found.size()!=0){
            accessXml.printBookDataBasedOfId(*std::next(found.begin(), 0));
            found.pop_front();
        }
```

Fungsi printBookDataBasedOfId tampak seperti dibawah, fungsi ini meminta argument berupa id dari buku yang hendak diprint datanya, didalam fungsi ini dilakukan loop antar node buku, apabila ditemukan kecocokan id, dilakukan std::cout untuk setiap metadata buku

```
    void printBookDataBasedOfId(std::string id){
        // Iterate over the book nodes
        for (rapidxml::xml_node<> * book_node = root_node->first_node("book");
                    book_node;
```

```
                    book_node = book_node->next_sibling())
        {
            // Iterate over the book data if the id is match
            if(book_node->first_attribute("id")->value()==id){
                std::cout << "| id      " << "= " << std::setw(100) <<
std::left << book_node->first_attribute("id")->value() << " |" << std::endl;
                for(rapidxml::xml_node<> * Book_metadata = book_node-
>first_node();
                        Book_metadata;
                        Book_metadata = Book_metadata->next_sibling())
                {
                    // std::cout everything except the link
                    std::string name_book_metadata = Book_metadata->name();
                    if(name_book_metadata == "price"){
                        std::string value = Book_metadata->value();
                        std::string price = "IDR " + value + ",00";
                        std::cout << "| " << std::setw(7) << std::left <<
Book_metadata->name() << "= " << std::setw(100) << std::left << price << " |"
<< std::endl;
                    }
                    else if(name_book_metadata != "link"){
                        std::cout << "| " << std::setw(7) << std::left <<
Book_metadata->name() << "= " << std::setw(100) << std::left << Book_metadata-
>value() << " |" << std::endl;
                    }
                }
                std::cout << "| " << std::setw(111) << std::right << " |" <<
std::endl;
            }
        }
    }
```

Setelah dilakukan print hasil pencarian, program meminta input dari user mengenai kode buku yang ingin dimasukkan keranjang, program akan terus meminta input apabila id tersebut tidak terdapat pada set yang dibuat pada awal program, loop baru akan berhenti jika user sudah memasukkan id yang valid atau memasukkan s/d

```
        //get it in the cart
        std::cout << "| Enter the product's ID that you want to buy\n| (input
's' to back to search or 'd' if you done with it): ";
        do{
            std::cin >> choose;
            std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
            if (all_book_ids.count(choose) || choose == "s" || choose == "d")
                break;

            std::cout << "| The code's is not valid" << std::endl;
        }while (true);
```

Setelah keluar dari loop, program akan memasukkan pilihan pada keranjang, jika user memasukkan ‘d’ maka user akan keluar dari fungsi customer, jika user memasukkan ‘s’ program akan melanjutkan ke iterasi selanjutnya tanpa memasukkan pilihan ke keranjang

```
        if(choose == "d"){
            return cart;
        }
        else if(choose == "s"){
            clear_screen();
            continue;
        }
        cart.push(choose);
```

Kemudian program akan memberikan rekomendasi mengenai buku yang serupa dengan buku yang dipilih oleh user sebelumnya, rekomendasi ini diambil dari graph yang digenerate diawal tadi

```
        std::cout << "| " << choose << " already added to the cart\n";

        //Print the list of recommendation based of similarities
        std::cout << "| Recommendation:\n";
        for (const auto& similarBook : recommendation){
            if(choose.compare(similarBook.book_a)==0){
                if(similarBook.similarity_type.compare("author")==0){
                    std::cout << "| " << similarBook.book_b << " - " << "same
author\n";
                }
                if(similarBook.similarity_type.compare("genre")==0){
                    std::cout << "| " << similarBook.book_b << " - " << "same
genre\n";
                }
            }
        }
        WaitEnter();
        clear_screen();
    }
```

Setelah dilakukan pengisian stack dengan fungsi customer, program akan memasukkan pointer atas objek kedalam map dan memasukkan orang tersebut kedalam antrian

```
                all_buyer[people] = Buyer;
                buyer_queue.push(people);
                ++people;
                break;
            }
```

Selanjutnya, user akan Kembali ke menu reroll

```
        clear_screen();
        bool canICloseTheStore = false;
        std::cout << "| === WELCOME TO BOOKSTORE SIMULATOR === |\n";
```

```
        std::cout << "| CHOOSE YOUR ROLE                          |\n";
        std::cout << "| 1] Customer                               |\n";
        std::cout << "| 2] Cashier                                |\n";
        std::cout << "| What's your role : ";
```

jika user memilih 2, maka program akan menjalnkan kode blok pada case 2. Disini program akan memberitahukan sekali lagi bahwa user sekarang sedang bertindak sebagai kasir

```
            case 2:
            {
                XMLDataHandler accessXml;
                std::cout << "| Now, you will act like a cashier who will
serve the customer's purchase.\n";
                WaitEnter();
                clear_screen();
```

kemudian program akan melakukan pemrosesan atas pembelian dari orang yang berada di antrian terdepan, pemrosesan dilakukan oleh fungsi processingPurchase

```
                processingPuchase(buyer_queue.front());
                WaitEnter();
                buyer_queue.pop();
                break;
```

fungsi processingPurchase tampak seperti dibawah, fungsi ini meminta argument berupa nomor antrian. Kemudian mengakses keranjang dari map all_buyer dengan key nomor antrian itu sendiri. Setelah keranjang diakses, dilakukan iterasi terhadap seluruh item di keranjang dan setiap iterasi dilakukan print data setiap buku dan dilakukan penjumlahan harga. Di akhir user akan diminta untuk memasukkan uang yang dimiliki customer dan kemudian akan mengeprint struknya

```
void processingPuchase(int queueNumber){
    XMLDataHandler accessXml;
    buyer Buyer = *all_buyer[queueNumber];
    std::stack<std::string> Cart = Buyer.cart;

    //The cart is empty
    if(Cart.size()==0){
        std::cout << "You didn't buy anything, what are you doing here?\n";
        delete all_buyer[queueNumber];
        return;
    }

    //Iterate over the cart
    int total = 0; int product = 1;
    while(Cart.size()!=0){
        accessXml.printBookDataBasedOfId(Cart.top());
        //accessXml.printBook(Cart.top());
        total += accessXml.getPrice(Cart.top());
        Cart.pop();
    }
    std::cout << "| The total price is IDR " << total << ",00\n";
```

```
    int money = 0;
    while(true){
        std::cout << "| How much is your money : "; std::cin >> money;
        if(money>=total){break;}
        std::cout << "| The money is not enough\n";
    }
    accessXml.printReceipt(Buyer.cart, money);
    WaitEnter();
}
```

namun jika antrian kosong, program akan menanyakan pada user apakah user ingin melihat rekap penjualan hari ini, jika ya, program akan melakukan iterasi pada seluruh isi map sekaligus mengeprint metadata buku yang dibeli oleh masing-masing pembeli. Kemudioan diakhir program akan menghapus objek dengan fungsi delete

```
                if(buyer_queue.size()==0){
                    char confirm;
                    std::cout << std::setw(112) << std::left << "| There is no
people in the line" << " |" << std::endl;
                    std::cout << "| Do you want to see the recap for today
(y/n): "; std::cin >> confirm;
                    if(confirm=='y'){
                        for(const auto& it : all_buyer){
                            std::cout << "Buyer #" << it.first << std::endl;
                            buyer *Buyer = all_buyer[it.first];
                            std::stack<std::string> Cart = Buyer->cart;
                            while(Cart.size()!=0){
                                accessXml.printBookDataBasedOfId(Cart.top());
                                Cart.pop();
                            }
                            delete Buyer;
                            std::cout << std::endl;
                        }
                    }
                    canICloseTheStore = true;
                    break;
                }
```

- Tabel penggunaan struktur data

No	STRUKTUR DATA	POIN	PERAN
1	List	5	1. Pembuatan edge list untuk graph 2. Menampung hasil pencarian pada fungsi search
2	Stack	5	Menampung kumpulan ID buku yang dipilih oleh user Ketika menjadi customer
3	Queue	5	Mengantrikan setiap customer, customer yang antri pertama akan dilayani pertama

4	Set	5	Menampung data seluruh ID buku
5	Map	5	Menampung seluruh data pembelian customer
6	Tree	15	Menampung seluruh data buku pada file XML, parsing file XML dilakukan dengan library pihak ketiga RapidXML https://rapidxml.sourceforge.net/index.htm
7	Graph	10	Menampung hubungan antar buku (kesamaan author atau kesamaan genre) untuk memberikan rekomendasi buku yang serupa kepada user Ketika menjadi customer
Total poin : 50			