**“Students Data Management”**

A picture containing screenshot, graphics, graphic design, logo

Description automatically generatedLAPORAN PROYEK AKHIR

MATA KULIAH COMP6362004 – DATA STRUCTURES

KELAS LA20

Oleh :

2602181364 – Alfin Syaghaf Rifai

2602090435 – Audric Nagata

2602207363 – Yassar Annabil

**Semester [GENAP] 2022-2023**

**MALANG**

**LEMBAR PERSETUJUAN PROYEK AKHIR**

**“Students Data Management”**

MATA KULIAH COMP6362004 – DATA STRUCTURES KELAS BB20

Semester [GENAP] <2022-2023>

Laporan akhir proyek ini adalah benar karya kami :

**(ALFIN SYAGHAF RIFAI) (AUDRIC NAGATA) (YASSAR ANNABIL)**

**2602181364** 2602090435 2602207363

**Malang, 14 Juni 2023**

**(NYOMAN WIRA PRASETYA, S.Kom., M.T., M. Sc.)**

D6494

**BAB I**

**BACKGROUND**

*Binary Search Tree* adalah struktur data berhirarki yang menggunakan pendekatan pembagian dan pengecekan berulang untuk mencari, memasukkan, dan menghapus elemen data dengan efisien. BST terdiri dari *node-node* yang memiliki anak kiri dan anak kanan, di mana anak kiri memiliki nilai yang lebih kecil daripada *root node*, dan anak kanan memiliki nilai yang lebih besar.

Program *Student Data Management* yang menggunakan BST dapat memberikan keuntungan dalam pengorganisasian data mahasiswa, dengan memungkinkan pencarian, penambahan, dan penghapusan data secara efisien. Dalam konteks manajemen data mahasiswa, BST dapat digunakan untuk menyimpan data seperti nomor identitas mahasiswa, nama, alamat, program studi, dan informasi lainnya.

Dengan pemahaman tentang BST dan manfaatnya dalam manajemen data mahasiswa, program *Student Data Management* menggunakan BST dapat membantu mempermudah pengelolaan data mahasiswa, meningkatkan efisiensi pencarian, pengurutan, penambahan, dan penghapusan data, serta mencegah duplikasi data yang tidak diinginkan.

**BAB II**

**LITERATURE REVIEW**

**Data Structure**

Data Structure/Struktur Data adalah penyimpanan yang digunakan untuk menyimpan dan mengorganisir data. Ini merupakan cara untuk mengatur data di komputer agar dapat diakses dan diperbarui dengan efisien.

Struktur data tidak hanya digunakan untuk mengorganisir data, tetapi juga untuk memproses, mengambil, dan menyimpan data. Terdapat berbagai jenis struktur data dasar dan lanjutan yang digunakan dalam hampir setiap program atau sistem perangkat lunak yang telah dikembangkan. Oleh karena itu, kita perlu memiliki pengetahuan yang baik tentang struktur data.

Data Structure dibagi menjadi 2 jenis, yaitu Data Structure Linear dan Non-Linear.

* Data Structure Linear: Struktur data di mana elemen data disusun secara berurutan atau linear, Pada struktur data linear, setiap elemen terhubung dengan elemen sebelumnya dan setelahnya. Contoh dari struktur data linear adalah array, stack, queue, linked list, dll.
* Data Structure Non-Linear: Struktur data di mana elemen data tidak ditempatkan secara berurutan atau linear. Pada struktur data non-linear, kita tidak dapat melintasi semua elemen dalam satu kali proses. Contoh dari struktur data non-linear adalah tree dan graph.

**Tree**

Tree merupakan salah satu bentuk struktur data non-linear yang menggambarkan hubungan yang bersifat hirarkis (hubungan one to many) antara elemen-elemen. Tree bisa definisikan sebagai kumpulan simpul/node dengan satu elemen khusus yang disebut Root dan node lainnya. Tree juga adalah suatu graph yang acyclic, simple, connected yang tidak mengandung loop.

**Binary Tree**

Binary Tree merupakan salah satu bentuk struktur data tidak linear yang menggambarkan hubungan yang bersifat hirarkis (hubungan one to many) antara elemen-elemen. Tree bisa didefinisikan sebagai kumpulan simpul/node dengan satu elemen khusus yang disebut Root dan node lainnya ( disebut subtree).  
Dalam tree terdapat jenis-jenis tree yang memiliki sifat khusus, diantaranya adalah binary tree.

Binary tree adalah suatu tree dengan syarat bahwa tiap node (simpul) hanya boleh memiliki maksimal dua subtree dan kedua subtree tersebut harus terpisah. Tiap node dalam binary treee boleh memiliki paling banyak dua child (anak simpul), secara khusus anaknya  dinamakan kiri dan kanan. Binary Tree merupakan himpunan  vertex-vertex yang terdiri dari 2 subtree (dengan disjoint) yaitu subtree kiri dan subtree kanan. Setiap vertex dalam binary tree mempunyai derajat keluar max = 2.

**Binary Search Tree**

Binary Search Tree (BST) merupakan tree yang terurut (ordered Binary Tree) yang memiliki kelebihan bila dibanding dengan struktur data lain. Diantaranya adalah proses pengurutan (sorting) dan pencarian (searching) dapat dilakukan bila data sudah tersusun dalam struktur data BST. Pengurutan dapat dilakukan bila BST ditelusuri (traversed) menggunakan metode in-order.  Agar data benar-benar tersusun dalam struktur data BST, dua aturan yang harus dipenuhi pada saat data diatur dalam BST adalah sebagai berikut:

1. Semua data di bagian kiri sub-tree dari node t selalu lebih kecil dari data dalam node t itu sendiri.

2. Semua data di bagian kanan sub-tree dari node t selalu lebih besar atau sama dengan data dalam   
node t.

**File Processing**

Dalam Ilmu Komputer, File Processing adalah cara untuk menyimpan, mengambil, dan memanipulasi data yang terdapat dalam berbagai file.

File digunakan untuk menyimpan berbagai dokumen. Semua file dikelompokkan berdasarkan kategorinya. Nama file sangat terkait satu sama lain dan diatur dengan baik untuk memudahkan akses ke file tersebut. Dalam file processing, jika seseorang perlu memasukkan, menghapus, mengubah, menyimpan, atau memperbarui data, orang tersebut harus mengetahui seluruh hierarki file tersebut.

**BAB III**

**PROGRAM DEFINITION**

**Flowchart:**

**A picture containing diagram, sketch, plan, technical drawing

Description automatically generated**

**Pseudo Code:**

START

Structure Student:

idNumber: integer

name: string

gender: character

className: string

phoneNumber: integer

left: Student pointer

right: Student pointer

Function createStudent(idNumber, name, gender, className, phoneNumber):

newStudent = Allocate memory for Student struct

newStudent->idNumber = idNumber

newStudent->name = name

newStudent->gender = gender

newStudent->className = className

newStudent->phoneNumber = phoneNumber

newStudent->left = NULL

newStudent->right = NULL

return newStudent

Function insertStudent(root, idNumber, name, gender, className, phoneNumber):

if root is NULL:

 return createStudent(idNumber, name, gender, className, phoneNumber)

if idNumber < root->idNumber:

root->left = insertStudent(root->left, idNumber, name, gender, className, phoneNumber)

else:

        root->right = insertStudent(root->right, idNumber, name, gender, className, phoneNumber)

return root

Function searchStudent(root, idNumber):

if root is NULL or idNumber is equal to root->idNumber:

       return root

if idNumber < root->idNumber:

return searchStudent(root->left, idNumber)

else:

      return searchStudent(root->right, idNumber)

Function saveStudentDataToFile(root, file):

if root is not NULL:

        saveStudentDataToFile(root->left, file)

        Write root's data to file

        saveStudentDataToFile(root->right, file)

Function loadStudentDataFromFile(root, file):

while there are more lines in the file:

Read data from file

root = insertStudent(root, idNumber, name, gender, className, phoneNumber)

    return root

Function addStudent(root):

Read student details from user

    root = insertStudent(root, idNumber, name, gender, className, phoneNumber)

    return root

Function searchStudentMenu(root):

Read idNumber from user

    result = searchStudent(root, idNumber)

    if result is NULL:

   Display "Student data not found"

    else:

Display result's data

Function updateStudent(root):

Read upIdNumber from user

    update = searchStudent(root, upIdNumber)

    if update is NULL:

    Display "Student data not found"

else:

    Read new details from user

  Update update's data

Function deleteStudent(root, idNumber):

if root is NULL:

        return root

if idNumber < root->idNumber:

        root->left = deleteStudent(root->left, idNumber)

else if idNumber > root->idNumber:

   root->right = deleteStudent(root->right, idNumber)

else:

        if root->left is NULL:

temp = root->right

       Free root

       return temp

else if root->right is NULL:

            temp = root->left

            Free root

            return temp

         temp = root->right

        while temp->left is not NULL:

            temp = temp->left

        Copy temp's data to root

        root->right = deleteStudent(root->right, temp->idNumber)

    return root

Function deleteStudentMenu(root):

    Read idNumber from user

    result = searchStudent(root, idNumber)

    if result is NULL:

        Display "Student data not found"

    else:

        root = deleteStudent(root, idNumber)

Function printStudent(student):

    Display student's data

Function printStudentData(root):

    if root is not NULL:

        printStudentData(root->left)

        printStudent(root)

        printStudentData(root->right)

Function displayMenu():

    Display menu options

    Read user's choice

    return choice

main():

    root = NULL

     file = Open file

    root = loadStudentDataFromFile(root, file)

    Close file

    loop until user chooses to exit:

      choice = displayMenu()

if choice is 1

            root = addStudent(root)

      else if choice is 2:

           updateStudent(root)

      else if choice is 3:

          searchStudentMenu(root)

     else if choice is 4:

         deleteStudentMenu(root)

     else if choice is 5:

printStudentData(root)

 else if choice is 6:

Exit the loop

END

**Program Description**

Program dimulai dengan user diberikan 6 pilihan sebagai fitur utama program, yaitu *Add new student, Update student data, Search for student data, Delete student data, Print student data,* dan *exit*.

1. Add new student.  
     
   Pada fitur ini, user diminta untuk memasukkan data-data berupa Nomor Induk Mahasiswa(NIM), nama mahasiswa dengan batas 50 karakter, gender (berupa M/F), kelas, serta nomor telepon.
2. Update student data  
     
   Fitur ini memungkinkan user untuk mengubah data yang sudah dimasukkan melewati fitur add new student. User akan diminta untuk memasukkan ulang sekaligus mengganti data yang sudah terlanjur dituliskan dalam add new student.
3. Search for student data  
     
   Pada fitur ini, akan dilakukan pencarian berdasarkan NIM. Pencarian akan dilakukan dengan membandingkan NIM yang telah diinput dengan node-node dalam Binary Search Tree(BST), sesuai dengan karakteristik BST.
4. Delete student data  
     
   Fitur ini berfungsi untuk menghilangkan data salah satu mahasiswa yang sudah tercatat. Fitur delete ini menggunakan NIM untuk melakukan pencarian, lalu data akan di delete. Apabila data yang diinginkan untuk di delete tidak ada, maka program tidak akan menjalankan deletion node.
5. Print student data  
     
   Pada print student data, akan ditampilkan hasil data-data yang sudah diinput melalui add new student. Program akan menampilkan sebuah tabel berisi NIM, nama, gender, kelas, dan nomor telepon. Tabel itu akan disusun sesuai dengan NIM.
6. Exit  
     
   Fitur ini akan mengakhiri terminal secara langsung.

**Layout Design**

**A screen shot of a computer

Description automatically generated with medium confidence**

**A picture containing text, font, screenshot

Description automatically generated**

**A screen shot of a computer

Description automatically generated with medium confidence**

**A picture containing text, screenshot, font, black

Description automatically generated**

**A black screen with white text

Description automatically generated with low confidence**

**A screen shot of a computer

Description automatically generated with medium confidence**

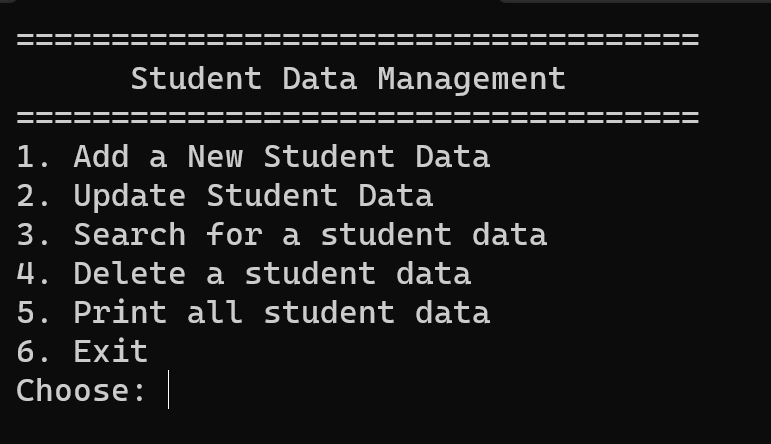
**Program Features**

Fitur yang ada pada program kami ini ada 5 yaitu *Insert New Student Data, Update Student Data, Search Student Data, Delete a Student Data,* dan *View List of All Student Data.*

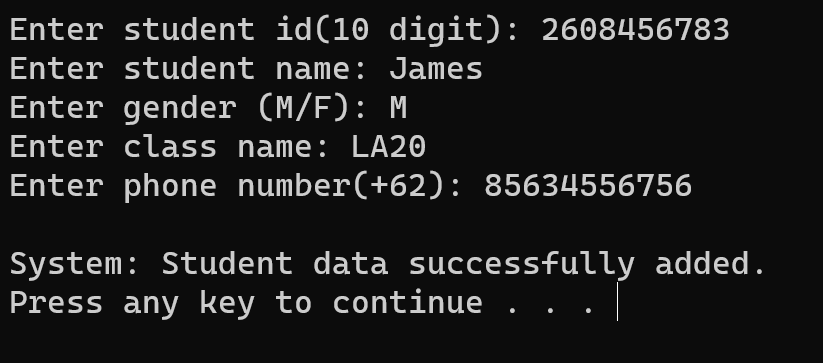
**BAB IV**

**RESULT**

**Program Screenshot and Explanation**

****

Tampilan awal pada saat program dijalankan.

****

Tampilan pada menu *Add a New Student Data,* *user* diminta untuk memasukkan beberapa data seperti *student id/*nim, nama mahasiswa, jenis kelamin, nama kelas, dan nomor telepon. Setelah *user* selesai memasukkan semua data tersebut system akan memasukkan ke dalam *Binary Search Tree* dan disimpan di *File Processing.*

**A screen shot of a computer

Description automatically generated with medium confidence**

Tampilan pada menu *Update Student Data,* *user* diminta untuk memasukkan *student id/*nim untuk dicari terlebih dahulu apakah data yang ingin diperbarui ada atau tidak. Jika data ditemukan maka *user* diminta untuk memasukkan kembali data yang ingin diperbarui seperti nama mahasiswa, jenis kelamin, nama kelas, dan nomor telepon. Setelah *user* selesai memasukkan semua data tersebut system akan memasukkan ke dalam *Binary Search Tree* dan disimpan di *File Processing.*

**A picture containing text, screenshot, font, black

Description automatically generated**

Tampilan pada menu *Search for a student data,* *user* diminta untuk memasukkan *student id/*nim untuk dicari terlebih dahulu apakah data ada atau tidak. Jika data ditemukan maka program akan menampilkan data mahasiswa yang dicari mulai dari nim, nama mahasiswa, jenis kelamin, nama kelas, dan nomer telepon.

**A black screen with white text

Description automatically generated with low confidence**

Tampilan pada menu *Delete Student Data,* *user* diminta untuk memasukkan *student id/*nim untuk dicari terlebih dahulu apakah data yang ingin dihapus ada atau tidak. Jika data ditemukan maka program akan menghapus data tersebut dari *Binary Search Tree* dan *File Processing* pada program.

**A screen shot of a computer

Description automatically generated with medium confidence**

Tampilan pada menu *Print All Student Data,* saat memilih menu ini program akan langsung memperlihatkan seluruh data yang tersimpan pada *Binary Search Tree* dan *File Processing* dan ditampilkan secara *InOrder Traversal.*

**Program Code:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

/\*

Project Name: Student Data Management

Member:

1. Alfin Syaghaf Rifai - 2602181364

2. Audric Nagata - 2602090435

3. Yassar Annabil - 2602207363

\*/

//structure data

typedef struct Student {

    long long int idNumber;

    char name[50];

    char gender;

    char className[20];

    long long int phoneNumber;

    struct Student\* left;

    struct Student\* right;

} Student;

// Fungsi untuk membuat data baru mahasiswa

Student\* createStudent(long long int idNumber, char name[], char gender, char className[], long long int phoneNumber) {

    Student\* newStudent = (Student\*)malloc(sizeof(Student));

    newStudent->idNumber = idNumber;

    strcpy(newStudent->name, name);

    newStudent->gender = gender;

    strcpy(newStudent->className, className);

    newStudent->phoneNumber = phoneNumber;

    newStudent->left = NULL;

    newStudent->right = NULL;

    return newStudent; //return struct pointer NewStudent

}

// Insertion data ke dalam BST

Student\* insertStudent(Student\* root, long long int idNumber, char name[], char gender, char className[], long long int phoneNumber) {

    if (root == NULL) {

        return createStudent(idNumber, name, gender, className, phoneNumber);

    }

    if (idNumber < root->idNumber) {

        root->left = insertStudent(root->left, idNumber, name, gender, className, phoneNumber);

    } else {

        root->right = insertStudent(root->right, idNumber, name, gender, className, phoneNumber);

    }

    return root;

}

Student\* searchStudent(Student\* root, long long int idNumber) {

    if (root == NULL || idNumber == root->idNumber) {

        return root;

    }

    if (idNumber < root->idNumber) {

        return searchStudent(root->left, idNumber);

    } else {

        return searchStudent(root->right, idNumber);

    }

}

// save data mahasiswa ke dalam textfile

void saveStudentDataToFile(Student\* root, FILE\* file) {

    if (root != NULL) {

        saveStudentDataToFile(root->left, file);

        fprintf(file, "%lld %s, %c %s, %lld\n", root->idNumber, root->name, root->gender, root->className, root->phoneNumber);

        saveStudentDataToFile(root->right, file);

    }

}

// load data mahasiswa dari dalam textfile

Student\* loadStudentDataFromFile(Student\* root, FILE\* file) {

    long long int idNumber;

    char name[50];

    char gender;

    char className[20];

    long long int phoneNumber;

    while (fscanf(file, "%lld %[^,], %c %[^,], %lld\n", &idNumber, name, &gender, className, &phoneNumber) != EOF) {

        root = insertStudent(root, idNumber, name, gender, className, phoneNumber);

    }

    return root;

}

Student\* addStudent(Student\* root) {

    long long int idNumber;

    char name[50];

    char gender;

    char className[20];

    long long int phoneNumber;

    printf("Enter student id(10 digit): ");

    scanf("%lld", &idNumber); getchar();

    printf("Enter student name: ");

    scanf("%[^\n]", name); getchar();

    printf("Enter gender (M/F): ");

    scanf("%c", &gender); getchar();

    printf("Enter class name: ");

    scanf("%[^\n]", className); getchar();

    printf("Enter phone number(+62): ");

    scanf("%lld", &phoneNumber); getchar();

    root = insertStudent(root, idNumber, name, gender, className, phoneNumber);

    printf("\nSystem: Student data successfully added.\n");

    system("pause");

    system("cls");

    return root;

}

void searchStudentMenu(Student\* root) {

    long long int idNumber;

    printf("Enter the id of the student\n");

    printf("Example: 2602346785\n");

    printf("=> ");

    scanf("%lld", &idNumber); getchar();

    Student\* result = searchStudent(root, idNumber);

    if (result == NULL) {

        printf("\nSystem: Student data not found.\n");

        system("pause");

        system("cls");

    } else {

        system("cls");

        printf("||======================| Student Data Found |======================\n");

        printf("|| Student ID   : %lld\n", result->idNumber);

        printf("|| Name     : %s\n", result->name);

        printf("|| Gender   : %c\n", result->gender);

        printf("|| Class    : %s\n", result->className);

        printf("|| Phone Number : %lld\n", result->phoneNumber);

        printf("||==================================================================\n");

        printf("\n");

        system("pause");

        system("cls");

    }

}

void updateStudent(Student\* root) {

    long long int upIdNumber;

    char newName[50];

    char newGender;

    char newClassName[20];

    long long int newPhoneNumber;

    printf("Enter student id you want to update\n");

    printf("Example: 2602346785\n");

    printf("=> ");

    scanf("%lld", &upIdNumber); getchar();

    Student\* update = searchStudent(root, upIdNumber);

    if (update == NULL) {

        printf("\nSystem: Student data not found.\n");

        system("pause");

        system("cls");

    } else {

        printf("Enter new name: ");

        scanf("%[^\n]", newName); getchar();

        printf("Enter new gender (M/F): ");

        scanf("%c", &newGender); getchar();

        printf("Enter new class: ");

        scanf("%[^\n]", newClassName); getchar();

        printf("Enter new phone number(+62): ");

        scanf("%lld", &newPhoneNumber); getchar();

        printf("\nSystem: Student data updated successfully.\n");

        system("pause");

        system("cls");

        update->idNumber = upIdNumber;

        strcpy(update->name, newName);

        update->gender = newGender;

        strcpy(update->className, newClassName);

        update->phoneNumber = newPhoneNumber;

    }

}

Student\* deleteStudent(Student\* root, long long int idNumber) {

    if (root == NULL) {

        return root;

    }

    if (idNumber < root->idNumber) {

        root->left = deleteStudent(root->left, idNumber);

    } else if (idNumber > root->idNumber) {

        root->right = deleteStudent(root->right, idNumber);

    } else {

        // kondisi tidak punya child atau hanya punya 1 child

        if (root->left == NULL) {

            Student\* temp = root->right;

            free(root);

            return temp;

        } else if (root->right == NULL) {

            Student\* temp = root->left;

            free(root);

            return temp;

        }

        // kondisi punya 2 child

        Student\* temp = root->right;

        while (temp->left != NULL) {

            temp = temp->left;

        }

        root->idNumber = temp->idNumber;

        strcpy(root->name, temp->name);

        root->gender = temp->gender;

        strcpy(root->className, temp->className);

        root->phoneNumber = temp->phoneNumber;

        root->right = deleteStudent(root->right, temp->idNumber);

    }

    return root;

}

Student\* deleteStudentMenu(Student\* root) {

    long long int idNumber;

    printf("Enter student id you want to delete\n");

    printf("Example: 2602346785\n");

    printf("=> ");

    scanf("%lld", &idNumber); getchar();

    Student\* result = searchStudent(root, idNumber);

    if (result == NULL) {

        printf("System: Student data not found.\n");

        system("pause");

        system("cls");

    } else {

        root = deleteStudent(root, idNumber);

        printf("\nSystem: Student data successfully deleted.\n");

        system("pause");

        system("cls");

    }

    return root;

}

// untuk print data-data mahasiswanya

void printStudent(Student\* student) {

    printf("|| %-14lld | %-25s |   %c    | %-4s  | %-12lld   ||\n", student->idNumber, student->name, student->gender, student->className, student->phoneNumber);

}

// Fungsi yang akan dipanggil saat print seluruh data mahasiswa

void printStudentData(Student\* root) {

    if (root != NULL) {

        //print menggunakan Pre-Order Traversal

        printStudentData(root->left);

        printStudent(root);

        printStudentData(root->right);

    }

}

int main() {

    Student\* studentData = NULL;

    long long int idNumber;

    //read data dari textfile

    FILE\* file = fopen("studentdata.txt", "r");

    if (file != NULL) {

        studentData = loadStudentDataFromFile(studentData, file);

        fclose(file);

    }

    char choice;

    do {

        printf("====================================\n      Student Data Management\n====================================\n");

        printf("1. Add a New Student Data\n");

        printf("2. Update Student Data\n");

        printf("3. Search for a student data\n");

        printf("4. Delete a student data\n");

        printf("5. Print all student data\n");

        printf("6. Exit\n");

        printf("Choose: ");

        scanf(" %c", &choice); getchar();

        switch (choice) {

            case '1': { //insert new data

                system("cls");

                studentData = addStudent(studentData);

                file = fopen("studentdata.txt", "w");

                if (file != NULL) {

                    saveStudentDataToFile(studentData, file);

                    fclose(file);

                } else {

                    printf("Error: Unable to open file.\n");

                    system("pause");

                    system("cls");

                }

                break;

            }

            case '2': { //update data

                system("cls");

                updateStudent(studentData);

                file = fopen("studentdata.txt", "r+");

                if (file != NULL) {

                    saveStudentDataToFile(studentData, file);

                    fclose(file);

                } else {

                    printf("Error: Unable to open file.\n");

                    system("pause");

                    system("cls");

                }

                break;

            }

            case '3': { //search student

                system("cls");

                searchStudentMenu(studentData);

                break;

            }

            case '4': { //delete student data

                system("cls");

                studentData = deleteStudentMenu(studentData);

                file = fopen("studentdata.txt", "w+");

                if (file != NULL) {

                    saveStudentDataToFile(studentData, file);

                    fclose(file);

                } else {

                    printf("Error: Unable to open file.\n");

                    system("pause");

                    system("cls");

                }

                break;

            }

            case '5': { //print

                system("cls");

                printf("||==============================================================================||\n");

                printf("||                               All Student Data                               ||\n");

                printf("||==============================================================================||\n");

                printf("|| Student Number | Name                      | Gender | Class | Phone Number   ||\n");

                printStudentData(studentData);

                printf("||==============================================================================||\n");

                printf("\n");

                system("pause");

                system("cls");

                break;

            }

            case '6': { //exit

                printf("\nQuitting the program...\n");

                system("pause");

                system("cls");

                printf("System: Thank you. Have a nice day!\n");

                break;

            }

            default: {

                printf("System: Invalid choice. Choose the number that on the menu.\n");

                system("pause");

                system("cls");

            }

        }

    } while (choice != '6');

    return 0;

}

**BAB V**

**REFERENCES**

1. GeeksforGeeks. (2023). *Data Structure*. Retrieved from: <https://www.geeksforgeeks.org/data-structures/>
2. Abidin, T. F. *Binary Search Tree*. Retrieved from: <https://informatika.unsyiah.ac.id/irvanizam/teaching/SD/bst.pdf>
3. Saxena, S. (2022, 4 Maret). *File Porcessing System.* Retrieved from: <https://www.geeksforgeeks.org/file-processing-system-fps/>

LEMBAR PENILAIAN

**“Program Students Data Management menggunakan Bahasa C”**

MATA KULIAH COMP6362004 – DATA STRUCTURES KELAS BB20

Semester [ Genap] 2022-2023

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **DAFTAR MAHASISWA** | **NILAI** | | | | **BOBOT** | | | | **KREDIT** | | | | **TOTAL KREDIT** |
| **1** | **2** | **3** | **4** | **1** | **2** | **3** | 4 | 1 | 2 | 3 | 4 |
| 2602181364 – Alfin Syaghaf Rifai |  |  |  |  | 20% | 30% | 30% | 20% |  |  |  |  |  |
| 2602090435 – Audric Nagata |  |  |  |  | 20% | 30% | 30% | 20% |  |  |  |  |  |
| 2602207363 – Yassar Annabil |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **TOTAL** | | | | | | | | | | | | |  |

# KETERANGAN :

## Skala Penilaian : 0 sd 100

* **Komponen**

|  |  |  |
| --- | --- | --- |
| 1 | : | Laporan |
| 2 | : | Produk |
| 3 | : | Pengetahuan dan Solusi |
| 4 | : | Presentasi |

**Malang, Minggu – 18 Juni 2023**

**(NYOMAN WIRA PRASETYA, S.Kom., M.T., M. Sc.)**

D6494

**BAB VI**

**OTHERS**

**Job Desc:**

2602181364 – Alfin Syaghaf Rifai

* Membuat function Insertion data, view list of all student, searching student.
* Merapihkan tampilan program
* Membuat Laporan

2602090435 – Audric Nagata

* Membuat function delete data
* Merapihkan tampilan program
* Membuat Laporan

2602207363 – Yassar Annabil

* Membuat function update data
* Merapihkan tampilan program
* Membuat laporan

Penilaian:

2602181364 – Alfin Syaghaf Rifai : 90

2602090435 – Audric Nagata : 90

2602207363 – Yassar Annabil : 90