LAPORAN STRUKTUR DATA

PENERAPAN *BINARY SEARCH TREE* (BST)

DALAM PENCARIAN DATA

Disusun oleh :

KELOMPOK 9

PUTU ANANDA PRADNYA WIRAWAN (1901010011)

I MADE HARY MAHAYANA (1901010046)

MUHAMMMAD AGUS ALFAN SALIM HAMID (1901010033)

PROGRAM STUDI ILMU KOMPUTER

FAKULTAS TEKNIK DAN DESAIN

UNIVERSITAS BUMIGORA

TAHUN PEMBELAJARAN 2019/2020

Jl. Ismail Marzuki No.22, Cilinaya, Kec. Cakranegara, Kota Mataram, Nusa Tenggara Barat, 83127, Indonesia.

Telephone : (0370) 638369, Email: kontak@universitasbumigora.ac.id

KATA PENGHANTAR

Assallamu'alaikum Warahmatullahi Wabarakatuh

Puji syukur khadirat TUHAN YME yang telah melimpahkan segala rahmat dan karunia-Nya, sehingga kami dapat menyelesaikan tugas Projek Struktur Data yang menjadi salah satu syarat untuk menyelesaikan Tugas Akhir Matakuliah Struktur Data di Universitas Bumigora Mataram.

Kami menyadari bahwa dalam peroses pengerjaan projek ini, kami tidak lepas dari peran berbagai pihak yang telah memberikan dukungan, bantuan dan dorongan sehingga projek ini dapat selesai. Dalam kesempatan ini, kami ingin mengucapkan terimakasih kepada Ibu Ni Gusti Ayu Dasriani, M.Kom. Selaku Dosen Matakuliah Sturktur Data.

Namun Terlepas dari semua itu, kami menyadari sepenuhnya bahwa masih ada kekurangan baik dari segi susunan kalimat maupun tata bahasanya serta keterbatasan pengetahuan dan pengalaman kami. Apabila di dalam makalah kami ini terdapat hal-hal yang dianggap tidak berkenan di hati pembaca mohon dimaafkan.

Wassalamu'alaikum Warahmatullahi Wabarakatuh.

Mataram, 2 Agustus 2020

(KELOMPOK 9)

DAFTAR ISI

BAB I PENDAHULUAN

1. Latar Belakang

Binary Search Tree adalah sebuah konsep penyimpanan data, dimana data disimpan dalam bentuk tree yang terurut (ordered Binary Tree), setiap node dapat memiliki anak maksimal 2 node. Memiliki kelebihan bila dibanding dengan struktur data lain yaitu kemudahan proses pengurutan dan pencarian.

Apabila data sudah tersusun dalam BST. Data dibagi menjadi dua dengan mencari titik tengah sebagai patokan, biasa disebut root. Kemudian dari root terdapat bagian kiri dan bagian kanan.

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 dibagian kiri sub-tree dari node t selalu lebih kecil dari data dalam node t itu sendiri.
2. Semua data dibagian kanan sub-tree dari node t selalu lebih besar atau sama dengan data dalam node t.

Adapun program sederhana yang dibuat adalah **Program Perpustakaan ILKOM A,** program ini dibuat dengan tujuan memudahkan proses pengurutan serta pencarian data buku. program tersebut dibangun menggunakan aplikasi DEV C++ dan menggunakan Bahasa C.

2. Tujuan

Adapun tujuan dibuatnya makalah ini antara lain :

1. Untuk memenuhi tugas project akhir mata kuliah Struktur Data
2. Mengeimplementasikan Mata kuliah Struktur Data ke dalam bentuk Program Sederhana.

3. Rumusan Masalah

1. Jelaskan Proses pembuatan program perpustakaan ILKOM A menggunakan DEV C++ serta metode BST.

BAB II PEMBAHASAN

1. Analisa

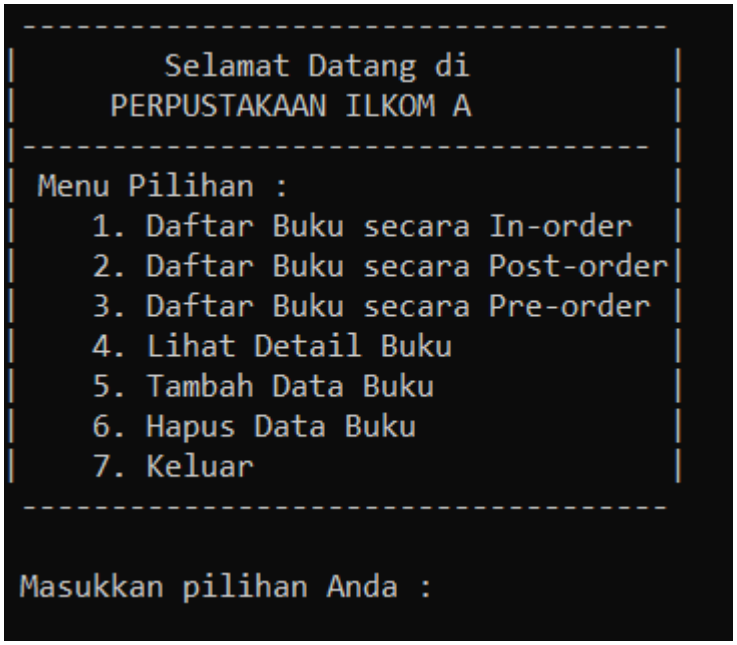
Analisa Dilakukan dengan mengidentifikasi masalah yang terjadi dalam aplikasi yang akan di bangun. Dalam hal ini, Aplikasi ini di maksudkan untuk memepermudah proses pengurutan data serta pencarian data secara efisien. Aplikasi ini dibangun menggunakan Software DEV C++ menggunakan bahasa C dengan Metode Struktur Data *Binary Search Tree* (BST) .

2. Impleentasi program

Ada 2 komponen utama yang paling penting sebelum membuat program, yakni:

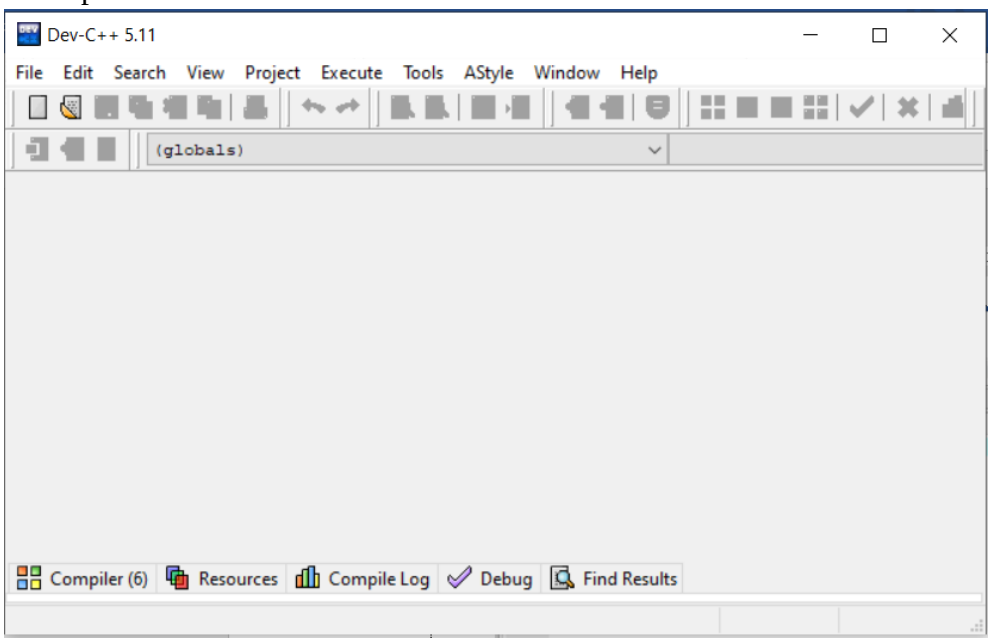
a. Laptop
b. Software Dev C++

Adapun Output Program yang di harapkan seperti dibawah ini.



Proses pembuatan program

1. Tampilan awal software DEV C++



2. Pendeklarasian header file dan Pendeklarasian tree awal.
   2.1 Pendeklarasian Struct.
   2.2 Pendeklamsian variabel yang akan digunakan untuk melakukan penyimpanan dan pemrosesan data, seta digunakan sebagai pammeter fungsi.
   2.3 Pendeklarasian Node *pohon bemilai NULL

perpustakaan ilkom a.cpp

```
1   #include <stdio.h>
2   #include <conio.h>
3   #include <stdlib.h>
4   #include <iostream>
5   #include <iomanip>
6   #include <string.h>
7   using namespace std;
8
9   //pendeklarasian tree
10  struct Node
11  {
12      int data,tahun;
13      char *judul1[11][20];
14      char *penulis[11][20];
15      Node *kiri;
16      Node *kanan;
17  };
18  Node *pohon = NULL;
19
```

3. Pendeklarasian fungsi untuk menambahkan node baru.
   3.1 Jika root masih kosong.
      3.1.1 Pembuatan node baru.
      3.1.2 Pengalokasian memori dari node yang telah dibuat.
      3.1.3 Inisialisasi awal node bam.
   3.2 Jika data yang akan dimasukkan lebih kecil dari root, maka diletakkan di node sebelah kiri. Selanjutnya panggil fungsi tambah node baru (3. 1).
   3.3 Jika data yang akan dimasukkan lebih besar dari root, maka diletakkan di node sebelah kanan. Selanjutnya panggil fungsi tambah node baru (3.1).
   3.4 Jika data yang akan dimasukkan sama dengan root.

```
//fungsi untuk menambahkan node baru
void tambah (Node **root,int databaru,char judul2[20],int tahun1,char penulis1[20])
{
    //jika root masih kosong
    if ((*root) == NULL)
       {
        Node *baru;       //pembuatan node baru
        baru = new Node;  //pengalokasian memori dari node yang telah dibuat
        //inisialisasi awal node baru
        baru->data = databaru;
        baru->kiri = NULL;
        baru->kanan = NULL;
        baru->judul1[11][20] = &judul2[20];
        baru->tahun = tahun1;
        baru->penulis[11][20] = &penulis1[20];
        (*root) = baru;
        (*root)->kiri = NULL;
        (*root)->kanan = NULL;
        //jika menunjuk ke NULL artinya tidak mempunyai child
        cout<<"\n";
        cout<<" Data Bertambah!"<<endl;
       }

    //jika data yang akan dimasukkan lebih kecil dari root,
    //maka diletakkan di node sebelah kiri
    else if (databaru < (*root)->data)
        tambah(&(*root)->kiri,databaru,judul2,tahun1,penulis1);

    //jika data yang akan dimasukkan lebih besar dari root,
    //maka diletakkan di node sebelah kanan
    else if (databaru > (*root)->data)
        tambah(&(*root)->kanan,databaru,judul2,tahun1,penulis1);

    //jika data yang akan dimasukkan sama dengan root
    else if (databaru == (*root)->data)
        cout<<" Data Sudah ada!";
}
```

4. Pendeklarasian fungsi untuk mencetak tree secara in-Order.

```
//fungsi untuk mencetak tree secara inOrder, postorder, dan preorder
void inOrder(Node *root)
{       int k;
        if(root != NULL)
        {
            inOrder(root->kiri);
            if(root->data != NULL)
            {
                k=strlen(root->judul1[11][20]);
                cout<<"|   "<<root->data<<"   |";
                if ((root->data) >= 11000 && (root->data) <= 11999)
                        cout<<" "<<setw(10)<<root->judul1[11][20]<<setw(47-k)<<
                        "|  Buku Bacaan     |";
                else if((root->data) >= 12000 && (root->data) <= 12999)
                        cout<<" "<<setw(10)<<root->judul1[11][20]<<setw(47-k)<<
                        "|  Buku Pelajaran  |";
                cout<<endl;
            }
            inOrder(root->kanan);
         }
}
```

5. Pendeklarasian fungsi untuk mencetak tree secara Post-Order.

```
//fungsi yang digunakan untuk mencetak tree secara postOrder
void postOrder(Node *root)
{
    int k;
       if(root!=NULL)
       {
         postOrder(root->kiri);
         postOrder(root->kanan);
         if(root->data!=NULL)
         {
            k=strlen(root->judul1[11][20]);
               cout<<"|   "<<root->data<<"   |";
               if ((root->data) >= 11000 && (root->data) <= 11999)
                        cout<<" "<<setw(10)<<root->judul1[11][20]<<setw(47-k)<<
                        "|  Buku Bacaan     |";
               else if((root->data) >= 12000 && (root->data) <= 12999)
                        cout<<" "<<setw(10)<<root->judul1[11][20]<<setw(47-k)<<
                        "|  Buku Pelajaran  |";
               cout<<endl;
         }
       }
}
```

6. Pendeklarasian fungsi untuk mencetak tree secara Pre-Order.

```
//fungsi yang digunakan untuk mencetak tree secara pre-Order
void preOrder(Node *root)
{
    int k;
    if(root!=NULL)
    {
        if(root->data!=NULL)
        {
            k=strlen(root->judul1[11][20]);
              cout<<"|   "<<root->data<<"   |";
              if ((root->data) >= 11000 && (root->data) <= 11999)
                        cout<<" "<<setw(10)<<root->judul1[11][20]<<setw(47-k)<<
                        "|  Buku Bacaan     |";
              else if((root->data) >= 12000 && (root->data) <= 12999)
                        cout<<" "<<setw(10)<<root->judul1[11][20]<<setw(47-k)<<
                        "|  Buku Pelajaran  |";
              cout<<endl;
        }
        preOrder(root->kiri);
        preOrder(root->kanan);
    }
}
```

7. Pendeklarasian fungsi untuk melakukan pencarian data.

```
//fungsi untuk melakukan pencarian data
void search(Node **root, int cari)
{
    if((*root) == NULL)
        cout<<" Data tidak ditemukan!";

    //jika data yang dicari lebih kecil dari root
    else if(cari < (*root)->data)
        search(&(*root)->kiri , cari);

    //jika data yang dicari lebih besar dari root
    else if(cari > (*root)->data)
        search(&(*root)->kanan , cari);

    //jika data yang dicari sama dengan root
    else if(cari == (*root)->data)
    {
        cout<<endl;
        cout<<" Berikut adalah data buku dengan kode "<<cari<<endl;
        cout<<" Judul Buku   = "<<(*root)->judul1[11][20]<<endl;
        cout<<" Penulis Buku = "<<(*root)->penulis[11][20]<<endl;
        cout<<" Tahun Terbit = "<<(*root)->tahun<<endl;
    }
    else
    {
        system("cls");
        cout<<" Data tidak ditemukan!"<<endl;
    }
}
```

8. Pendeklarasian Fungsi untuk mengetahui ketinggian/kedalaman Tree

```
//Fungsi untuk mengetahui ketinggian/kedalaman
int height(Node *root)
{
        if(root == NULL)
                return -1;
        else{
         int u = height(root->kiri);
         int v = height(root->kanan);
              if(u > v)
              return u + 1;
              else
              return v + 1;
        }
}
```

9. Pendeklarasian fungsi yang digunakan untuk menghapus suatu node

```
//fungsi yang digunakan untuk menghapus suatu node
void hapus(Node **root, int del)
{
      Node *curr;
      Node *parent;
      curr = (*root);

      bool flag = false;
      while(curr != NULL)
      {
            if(curr->data == del)
            {
                  flag = true;
                  cout<<" ------------------------------------------------------------ "<<endl;
                  cout<<"                 Data berhasil dihapus                        "<<endl;
                  cout<<" ------------------------------------------------------------ "<<endl<<endl;
                  break;
            }
            else
            {
                parent = curr;
                if(del > curr->data)
                  curr = curr->kanan;
                else
                  curr = curr->kiri;
            }
      }
   if(!flag)
   {
       system("cls");
       cout<<" Data tidak ditemukan. Penghapusan tidak bisa dilakukan."<<endl;
       return;
   }
```

```
//hanya satu tingkat, dengan kata lain hanya terdapat root
//jika hanya terdapat root, maka curr node tidak punya parent
if(height(pohon) == 0)
{
    if(curr->kiri == NULL && curr->kanan == NULL)
    {
        (*root) = NULL;
        return;
    }
}

//lebih dari satu tingkat, sehingga node curr mempunyai parent
else if(height(pohon) > 0)
{
    //1. jika node yang dihapus tidak memiliki anak
    if(curr->kiri == NULL && curr->kanan == NULL)
    {
        //jika node merupakan anak kiri dari parent
        if(parent->kiri == curr)
        {
            //replace parent->kiri dengan NULL
            parent->kiri = NULL;
            delete curr;
        }
        else //jika node merupakan anak kanan dari parent
        {
            //replace parent->kanan dengan NULL
            parent->kanan = NULL;
            delete curr;
        }
        return;
    }
```

```
//2. Jika node memiliki anak tunggal (anak kiri/anak kanan)
if((curr->kiri == NULL && curr->kanan != NULL) ||
   (curr->kiri != NULL && curr->kanan == NULL))
{
    //jika curr memiliki anak tunggal di sebelah kanan
    if(curr->kiri == NULL && curr->kanan != NULL)
    {
        //jika curr(data yang ingin dihapus) merupakan anak
        //kiri dari parent
        if(parent->kiri == curr)
        {
            //ganti isi parent->kiri dengan curr->kanan
            parent->kiri = curr->kanan;
            delete curr;
        }
        else //jika curr(data yang ingin dihapus) bukan
            //merupakan anak kiri dari parent
        {
            //ganti isi parent->kanan dengan curr->kanan
            parent->kanan = curr->kanan;
            delete curr;
        }
    }
    else //jika curr memiliki anak tunggal di sebelah kiri
    {
        //jika curr(data yang ingin dihapus) merupakan anak
        //kiri dari parent
        if(parent->kiri == curr)
        {       //ganti isi parent->kiri dengan curr->kiri
            parent->kiri = curr->kiri;
            delete curr;
        }
```

```
            else //jika curr(data yang ingin dihapus)
                 //bukan merupakan anak kiri dari parent
            {
             //ganti isi parent->kanan dengan curr->kiri
                parent->kanan = curr->kiri;
                delete curr;
             }
        }
        return;
}

//3. Node dengan dua anak
//ganti node dengan nilai terkecil dari Sub Tree Kanan
if (curr->kiri != NULL && curr->kanan != NULL)
{
//variabel bantu ini digunakan agar posisi curr asli tidak berubah, (tetap pada posisi node yang akan dihapus)
//variabel bantu digunakan untuk mengarah ke suatu node
        Node* bantu;
        bantu = curr->kanan;

        //jika subtree kanan dari posisi node sekarang
        //(curr,node yang akan dihapus) tidak memiliki anak
        if((bantu->kiri == NULL) && (bantu->kanan == NULL))
        {
            //ganti node curr dengan bantu
            // sama dengan curr = (curr->kanan)->kanan
            curr = bantu;
            delete bantu;
            curr->kanan = NULL;
        }
        else //jika child kanan dari node curr memiliki child
        {
            //jika node child kanan dari curr memiliki child kiri
            if((curr->kanan)->kiri != NULL)
```

```
            //variabel bantu ini digunakan agar posisi curr asli tidak berubah, (tetap pada posisi node yang akan dihapus)
            //variabel bantu digunakan untuk mengarah ke
            //suatu node
            Node* bantu2;
            Node* bantu3;
            //berlaku sebagai parent dari bantu 2
            bantu3 = curr->kanan;         //!perhatikan
            bantu2 = (curr->kanan)->kiri; //!perhatikan

            //mengarahkan posisi node ke node terkiri(untuk
            // menuju ke node yang memiliki nilai terkecil)
            while(bantu2->kiri != NULL)
            {
                bantu3 = bantu2;
                bantu2 = bantu2->kiri;
            }
            //replace nilai dari node curr dengan nilai
            //dari node bantu
            curr->data = bantu2->data;
            delete bantu2;
            bantu3->kiri = NULL;
}
else //jika node child kanan dari curr tidak memiliki
    //child kiri
{
    Node* tmp;
    tmp = curr->kanan;
    //replace nilai dari node curr dengan nilai dari
    //node tmp (curr->kanan)
    curr->data = tmp->data;
    curr->kanan = tmp->kanan;
    delete tmp;
}
 }
return;
  }
}
}
```

10. Pendeklarasian fungsi Main Utama

```
//fungsi utama
int main()
{
   //inisialisasi data awal
   char pil,judulbku[20],penulis2[20];
   int tahun2, dell, cari, kode,i;
   char judulbuku[11][20]={"Struktur Data","Sistem Operasi","Bahasa Inggris",
                           "Matematika Diskrit","Kalkulus",
                          "Algoritma dan Pemrograman","Sistem Basis Data",
                          "Kecakapan Antar Personal","Bahasa Indonesia",
                          "Arsitektur Komputer"};
   int  tahunbuku[11]={2019,2017,2012,2018,2019,2015,2013,2010,2014,2019, };
   char pnlsbuku[11][20]={"Farida", "Hafifudin", "Tijani",
                          "Masgull","Apriansyah", "Juniadi",
                         "Intan", "Alfian",
                         "Abdimanaf", "Andisyahputra"};
   int  kode1[11]={11112,12331,11123,12345,11234,11221,12211,11541,12255,12555, };

   for(i=0;i<11;i++)
    tambah(&pohon, kode1[i], judulbuku[i], tahunbuku[i], pnlsbuku[i]);
```

```
ulangi:
 system("cls");
   cout<<" ------------------------------------ "<<endl;
 cout<<"|        Selamat Datang di           |"<<endl;
 cout<<"|     PERPUSTAKAAN ILKOM A           |"<<endl;
 cout<<"|------------------------------------ |"<<endl;
 cout<<"| Menu Pilihan :                     |"<<endl;
 cout<<"|    1. Daftar Buku secara In-order  |"<<endl;
 cout<<"|    2. Daftar Buku secara Post-order|"<<endl;
 cout<<"|    3. Daftar Buku secara Pre-order |"<<endl;
 cout<<"|    4. Lihat Detail Buku            |"<<endl;
 cout<<"|    5. Tambah Data Buku             |"<<endl;
 cout<<"|    6. Hapus Data Buku              |"<<endl;
 cout<<"|    7. Keluar                       |"<<endl;
 cout<<" ------------------------------------ "<<endl<<endl;
 cout<<" Masukkan pilihan Anda : ";
 cin>>pil;

 fflush(stdin); //mengosongkan buffering
 system("cls");
```

```
switch(pil)
  {
      //Tampilkan daftar buku
      case '1':
          if(pohon != NULL)
          {
              //panggil fungsi untuk mencetak data secara inOrder
              kop();
              inOrder(pohon);
          }
          else
              cout<<" Data masih Kosong!!!";
          cout<<" ------------------------------------------------------------ "<<endl;
          cout<<endl<<" ( Tekan tombol apapun untuk kembali ke menu ) ";
          _getch();
          break;

      case '2':
          if(pohon != NULL)
          {
              //panggil fungsi untuk mencetak data secara inOrder
              kop();
              postOrder(pohon);
          }
          else
              cout<<" Data masih Kosong!!!";
          cout<<" ------------------------------------------------------------ "<<endl;
          cout<<endl<<" ( Tekan tombol apapun untuk kembali ke menu ) ";
          _getch();
          break;

      case '3':
          if(pohon != NULL)
          {
              //panggil fungsi untuk mencetak data secara inOrder
              kop();
              preOrder(pohon);
          }
          else
              cout<<" Data masih Kosong!!!";
          cout<<" ------------------------------------------------------------ "<<endl;
          cout<<endl<<" ( Tekan tombol apapun untuk kembali ke menu ) ";
          _getch();
          break;

      //Lihat detail buku
      case '4':
          if(pohon != NULL)
          {
             kop();
             inOrder(pohon);
             cout<<" ------------------------------------------------------------ "<<endl<<endl;
             cout<<" Masukkan kode buku yang ingin dicari : ";
             cin>>cari;
             //panggil fungsi untuk melakukan pencarian sekaligus
             //menampilkan detail buku
             search(&pohon, cari);
             cout<<endl;
          }
          else
             cout<<endl<<" Masih kosong!";
          cout<<endl<<" ( Tekan tombol apapun untuk kembali ke menu )";
          _getch();
          break;
```

```
        //Tambah data buku
        case '5':
            tambah:
             cout<<" ==== Format Kode Buku ===="<<endl;
             cout<<"  Buku Bacaan     => 11xxx  "<<endl;
             cout<<"  Buku Pelajaran => 12xxx  "<<endl;
             cout<<" ========================="<<endl;
                cout<<" Masukkan Kode Buku : ";
             cin>>kode;
            if (kode >= 11000 && kode <= 11999)
               cout<<" Jenis -> buku bacaan"<<endl<<endl;
            else if(kode >= 12000 && kode <= 12999)
               cout<<" Jenis -> buku pelajaran "<<endl<<endl;
            else
            {
               system("cls");
               cout<<" !!! Kode buku salah, isikan sesuai format !!!"
                   <<endl<<endl;
               goto tambah;
            }
            cout<<" Tulis data buku baru!"<<endl;
            cout<<" Judul buku       = "; cin>>judulbku;
            cout<<" Nama penulis     = "; cin>>penulis2;
            cout<<" Tahun terbit     = "; cin>>tahun2;
            cout<<"\n";
            //panggil fungsi untuk menambah node berisi data pada tree
            tambah(&pohon,kode,penulis2,tahun2,penulis2);
            cout<<endl<<" ( Tekan tombol apapun untuk kembali ke menu )";
            _getch();
            break;
```

```
        //Hapus data buku
        case '6':
            if(pohon != NULL)
            {
                cout<<" Daftar Buku Sebelum Penghapusan:       "<<endl<<endl;
                kop();
                inOrder(pohon);
                cout<<" ------------------------------------------------------------ "<<endl<<endl;
                cout<<" PENGHAPUSAN DATA"<<endl;
                cout<<" Masukkan kode buku yang ingin dihapus: ";
                cin>>dell;
                cout<<endl;
                //panggil fungsi yang digunakan untuk melakukan
                //penghapusan pada suatu node
                hapus(&pohon , dell);
                 }
            else
                cout<<endl<<" Masih kosong!";
            cout<<endl<<" ( Tekan tombol apapun untuk kembali ke menu )";
            _getch();
            break;
```

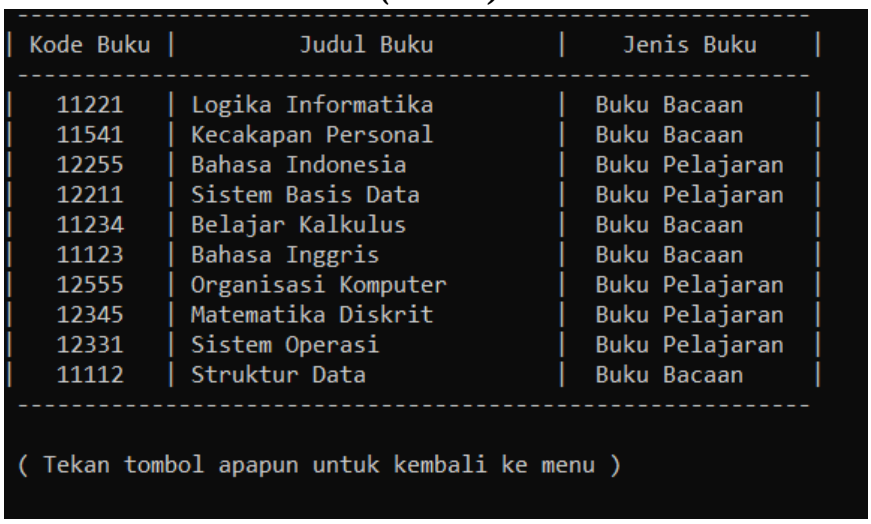
```
            //Keluar
            case '7':
                goto akhir;
                break;

            default:
                system("cls");
                goto ulangi;
        }
    goto ulangi;
getch();
akhir:
    return (0);
}
```

3. Running Program / Testing Program

Kode Buku	Judul Buku	Jenis Buku
11112	Struktur Data	Buku Bacaan
11123	Bahasa Inggris	Buku Bacaan
11221	Logika Informatika	Buku Bacaan
11234	Belajar Kalkulus	Buku Bacaan
11541	Kecakapan Personal	Buku Bacaan
12211	Sistem Basis Data	Buku Pelajaran
12255	Bahasa Indonesia	Buku Pelajaran
12331	Sistem Operasi	Buku Pelajaran
12345	Matematika Diskrit	Buku Pelajaran
12555	Organisasi Komputer	Buku Pelajaran

(Tekan tombol apapun untuk kembali ke menu)

(menu 1)



Kode Buku	Judul Buku	Jenis Buku
11221	Logika Informatika	Buku Bacaan
11541	Kecakapan Personal	Buku Bacaan
12255	Bahasa Indonesia	Buku Pelajaran
12211	Sistem Basis Data	Buku Pelajaran
11234	Belajar Kalkulus	Buku Bacaan
11123	Bahasa Inggris	Buku Bacaan
12555	Organisasi Komputer	Buku Pelajaran
12345	Matematika Diskrit	Buku Pelajaran
12331	Sistem Operasi	Buku Pelajaran
11112	Struktur Data	Buku Bacaan

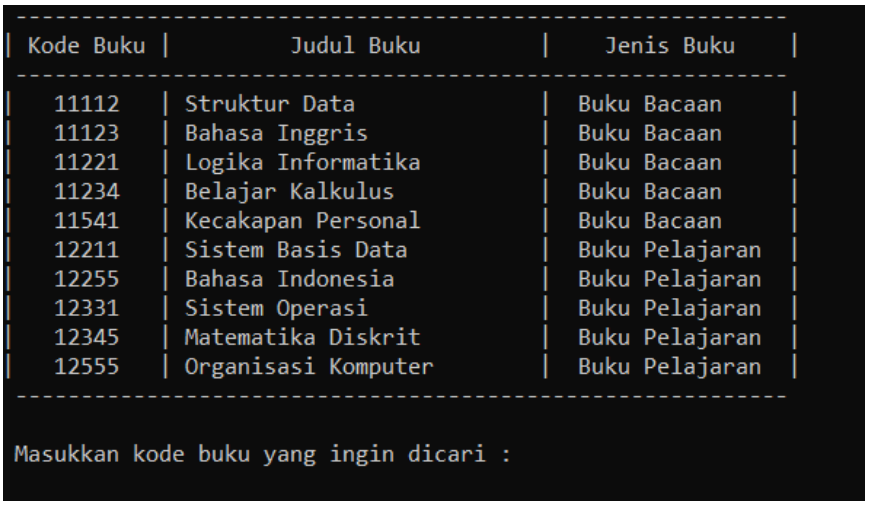
(Tekan tombol apapun untuk kembali ke menu)

(menu 2)

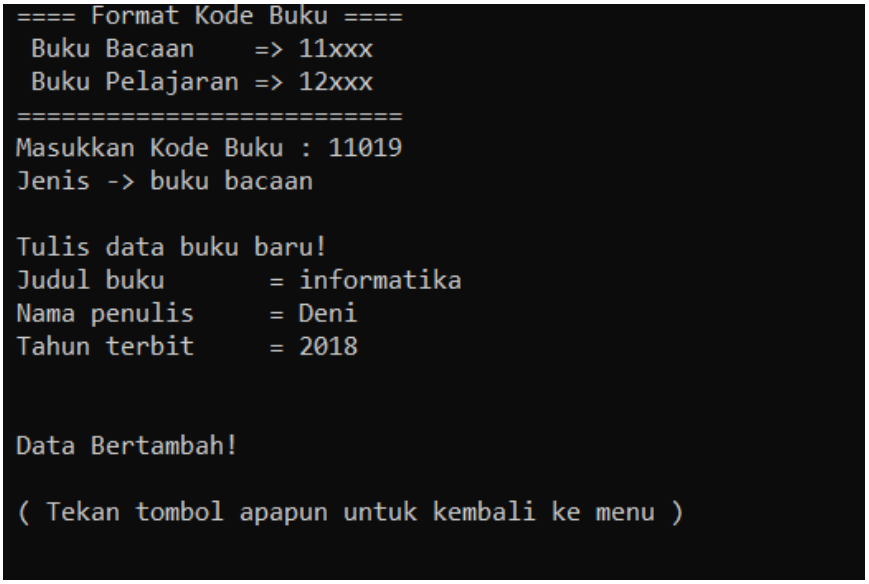
Kode Buku	Judul Buku	Jenis Buku
11112	Struktur Data	Buku Bacaan
12331	Sistem Operasi	Buku Pelajaran
11123	Bahasa Inggris	Buku Bacaan
11234	Belajar Kalkulus	Buku Bacaan
11221	Logika Informatika	Buku Bacaan
12211	Sistem Basis Data	Buku Pelajaran
11541	Kecakapan Personal	Buku Bacaan
12255	Bahasa Indonesia	Buku Pelajaran
12345	Matematika Diskrit	Buku Pelajaran
12555	Organisasi Komputer	Buku Pelajaran

(Tekan tombol apapun untuk kembali ke menu)

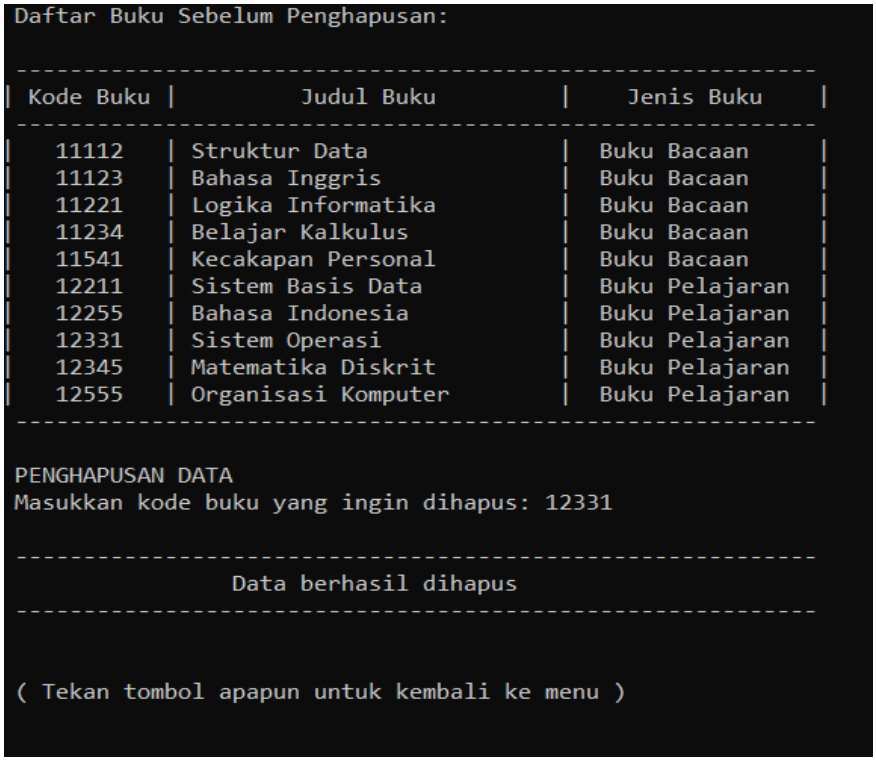
(menu 3)


Kode Buku	Judul Buku	Jenis Buku
11112	Struktur Data	Buku Bacaan
11123	Bahasa Inggris	Buku Bacaan
11221	Logika Informatika	Buku Bacaan
11234	Belajar Kalkulus	Buku Bacaan
11541	Kecakapan Personal	Buku Bacaan
12211	Sistem Basis Data	Buku Pelajaran
12255	Bahasa Indonesia	Buku Pelajaran
12331	Sistem Operasi	Buku Pelajaran
12345	Matematika Diskrit	Buku Pelajaran
12555	Organisasi Komputer	Buku Pelajaran
Masukkan kode buku yang ingin dicari :

(menu 4)


==== Format Kode Buku ====
Buku Bacaan => 11xxx
Buku Pelajaran => 12xxx
Masukkan Kode Buku : 11019
Jenis -> buku bacaan
Tulis data buku baru!
Judul buku = informatika
Nama penulis = Deni
Tahun terbit = 2018
Data Bertambah!
(Tekan tombol apapun untuk kembali ke menu)

(menu 5)


Daftar Buku Sebelum Penghapusan:
Kode Buku	Judul Buku	Jenis Buku
11112	Struktur Data	Buku Bacaan
11123	Bahasa Inggris	Buku Bacaan
11221	Logika Informatika	Buku Bacaan
11234	Belajar Kalkulus	Buku Bacaan
11541	Kecakapan Personal	Buku Bacaan
12211	Sistem Basis Data	Buku Pelajaran
12255	Bahasa Indonesia	Buku Pelajaran
12331	Sistem Operasi	Buku Pelajaran
12345	Matematika Diskrit	Buku Pelajaran
12555	Organisasi Komputer	Buku Pelajaran
PENGHAPUSAN DATA
Masukkan kode buku yang ingin dihapus: 12331
Data berhasil dihapus
(Tekan tombol apapun untuk kembali ke menu)

(menu 6)

BAB III PENUTUP

1. Kesimpulan

Binary Search Tree adalah sebuah konsep penyimpanan data, dimana data disimpan dalam bentuk tree yang terurut (ordered Binary Tree), setiap node dapat memiliki anak maksimal 2 node. Memiliki kelebihan bila dibanding dengan struktur data lain yaitu kemudahan proses pengurutan dan pencarian.

Adapun program sederhana yang dibuat adalah **Program Perpustakaan ILKOM A,** program ini dibuat dengan tujuan memudahkan proses pengurutan serta pencarian data buku. program tersebut dibangun menggunakan aplikasi DEV C++ dan menggunakan Bahasa C.

Ada 2 komponen utama yang paling penting sebelum membuat program tersebut, yakni:

c. Laptop
d. Software DEV C++

Terakhir, program ini memiliki fungsi Menu, atara lain ;

a. Daftar Buku secara In-Order
b. Daftar Buku secara Post-Order
c. Daftar Buku secara Pre-Order
d. Liat Detail Buku
e. Tambah Data Buku
f. Hapus Data Buku
g. Keluar

DAFTAR PUSTAKA

Referensi Laporan INDAH RAMADHANI.

http://informatika.unsyiah.ac.id/irvanizam/teaching/SD/bst.pdf diakses pada 2-8-2020.