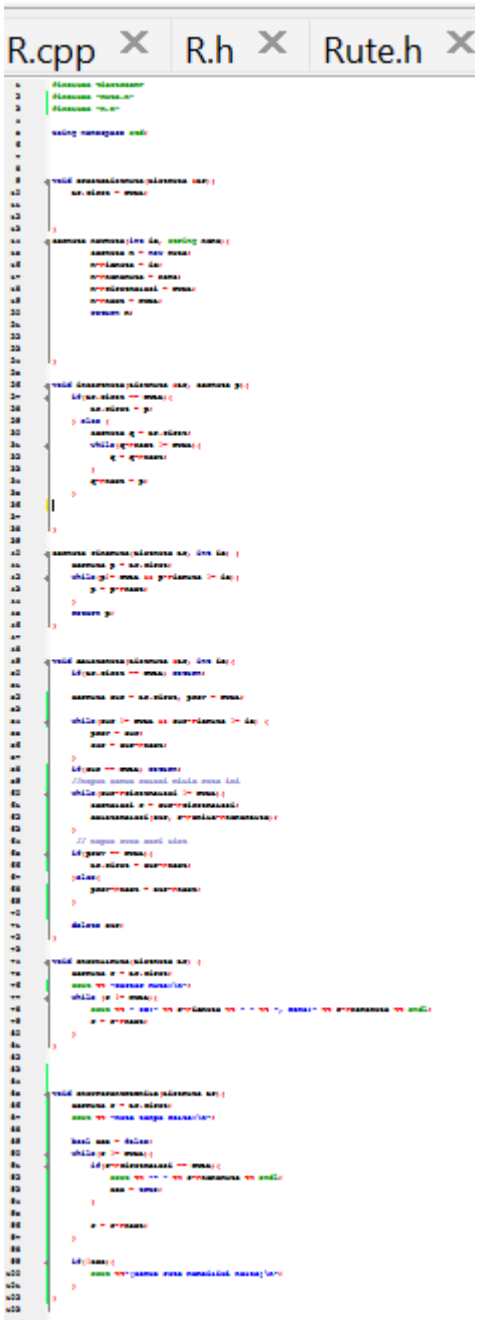
Kode 80%

Kelompok 15 , M.GHYMVANDYAR EL M. 103092400087 & LAURA DJASMIN TANDIAWAN

RUTE H DAN CPP

R.cpp | R.h | Rute.h | Rute.cpp | Halte.cpp

```h
#ifndef RUTE_H_INCLUDED
#define RUTE_H_INCLUDED

#include <iostream>
#include <string>
#include "R.h"
using namespace std;

struct Relasi;

struct Rute;
typedef Rute* adrRute;
typedef Relasi* adrRelasi;

struct Rute{
    int idRute;
    string namaRute;
    adrRelasi firstRelasi;
    adrRute next;

};

struct ListRute{
    adrRute first;
};



void createListRute(ListRute &Lr);
adrRute newRute(int id, string nama);
void insertRute(ListRute &Lr, adrRute r);
adrRute findRute(ListRute Lr, int id);
void deleteRute(ListRute &Lr, int id);


//tambahan responsi80
void showParentNoChild(ListRute Lr);
void showAllRute(ListRute Lr); // showAllParent









#endif // RUTE_H_INCLUDED

```

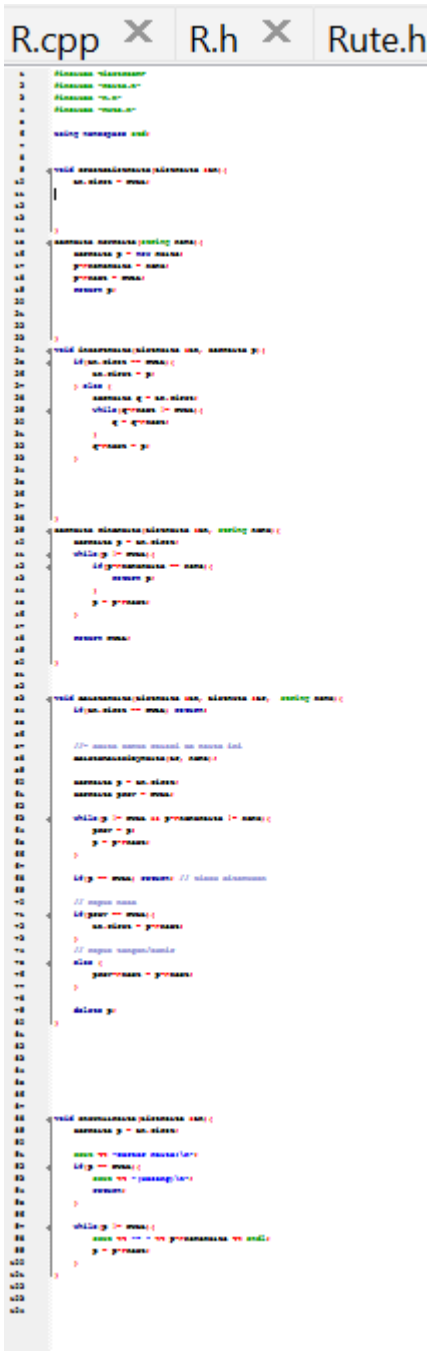


HALTE H DAN CPP

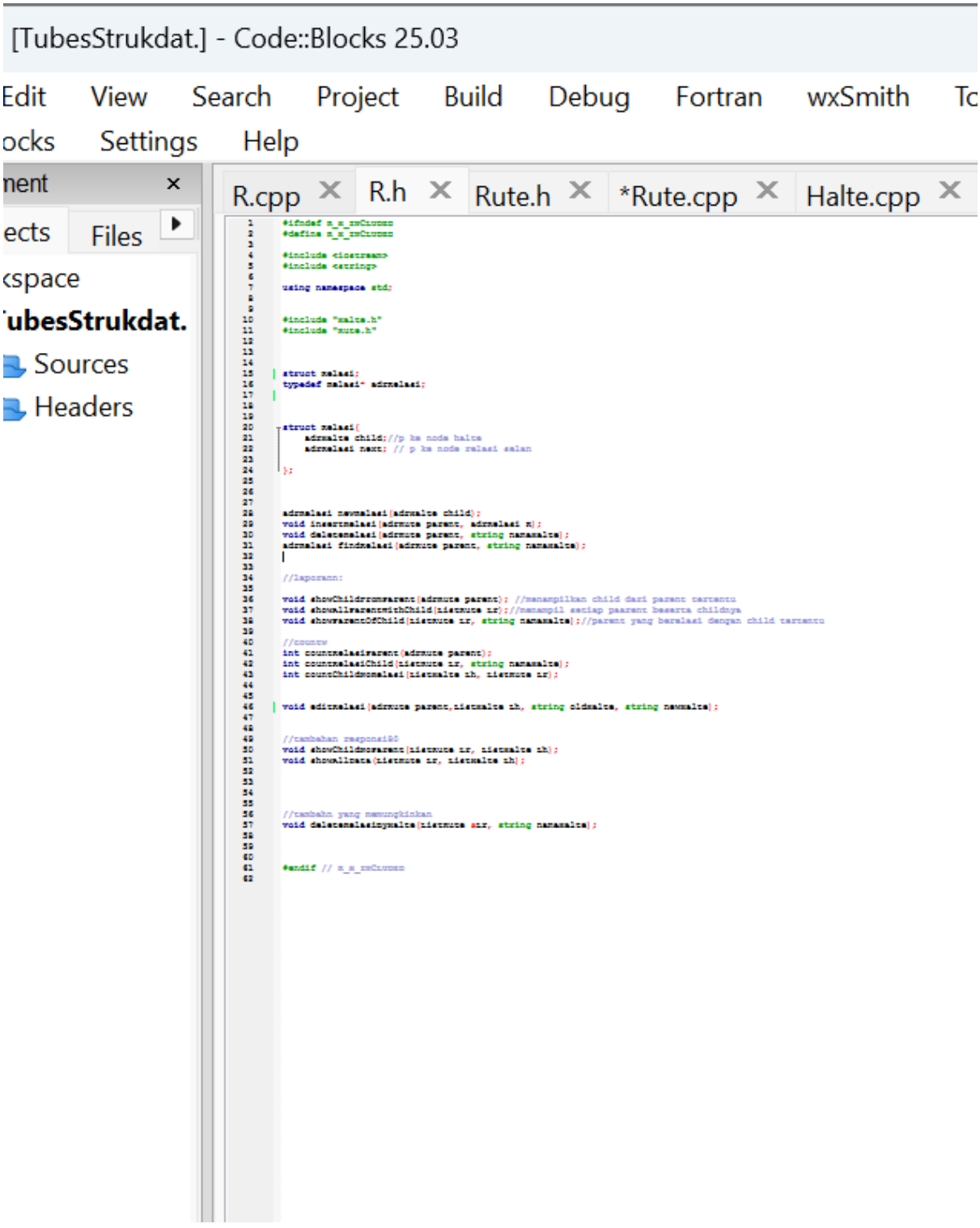
R.cpp | R.h | Rute.h | *Rute.cpp | Halte.cpp | Halte.h | main.cpp

D:\Desktop\vandy bahasa c\TubesStrukdat02\TubesStrukdat\R.cpp
Project: TubesStrukdat.

```
#include <iostream>
#include <string>
using namespace std;


// struct deklarasi
struct Halte;
typedef Halte* adrHalte;


struct Halte{
    string namaHalte;
    adrHalte next;


};

struct ListHalte {
    adrHalte first;

};


void createListHalte(ListHalte &Lh); // list kosong
adrHalte newHalte(string nama);// node halte baru
void insertHalte(ListHalte &Lh,adrHalte p);//
adrHalte findHalte(ListHalte &Lh, string nama);// serachir halte(child) be
void deleteHalte(ListHalte &Lh, ListRute &Lr, string nama);//hapus (by sea

//tambahan responsi80
void showAllHalte(ListHalte &Lh);//showAll halte









#endif // HALTE_H_INCLUDED

```



RELASI H DAN CPP


[TubesStrukdat.] - Code::Blocks 25.03
Edit View Search Project Build Debug Fortran wxSmith To
ocks Settings Help
nent
ects Files
kspace
ubesStrukdat.
Sources
Headers
R.cpp
R.h
Rute.h
*Rute.cpp
Halte.cpp
#ifndef R_H_INCLUDED
#define R_H_INCLUDED
#include <iostream>
#include <string>
using namespace std;
#include "halte.h"
#include "rute.h"
struct relasi;
typedef relasi* adrrelasi;
struct relasi{
adrhalte child;//p ke node halte
adrrelasi next; // p ke node relasi selan
};
adrrelasi newrelasi(adrhalte child);
void insertrelasi(adrrute parent, adrrelasi r);
void deleterelasi(adrrute parent, string namahalte);
adrrelasi findrelasi(adrrute parent, string namahalte);
//laporann:
void showChildrenofparent(adrrute parent); //menampilkan child dari parent tertentu
void showallparentwithChild(listrute lr);//menampil seciap paarent beserta childnya
void showparentOfChild(listrute lr, string namahalte);//parent yang berelasi dengan child tertentu
//counnw
int countrelasiparent(adrrute parent);
int countrelasiChild(listrute lr, string namahalte);
int countChildrenrelasi(listhalte hh, listrute lr);
void editrelasi(adrrute parent,listhalte hh, string oldhalte, string newhalte);
//tambahan responsi80
void showChildrenparent(listrute lr, listhalte hh);
void showallnata(listrute lr, listhalte hh);
//tambahn yang memungkinkan
void deleterelasimyhalte(listrute &lr, string namahalte);
#endif // R_H_INCLUDED

R.cpp | R.h | Rute.h | *Rute.cpp

```cpp
#include<iostream>
#include "R.h"
using namespace std;


drmalasi newmalasi(adrmalte child){
    adrmalasi  n = new malasi;
    n-> child = child;
    n-> next= NULL;
    return n;


void insertmalasi(adrmute parent, adrmalasi n){
    if(parent->firstmalasi == NULL){
        parent->firstmalasi = n;
    }else{
        adrmalasi p = parent->firstmalasi;
        while(p->next != NULL){
            p = p->next;
        }
        p->next = n;
    }


drmalasi findmalasi(adrmute parent, string namamalte){
    adrmalasi p = parent->firstmalasi;
    while(p != NULL){
        if(p->child->namamalte == namamalte){
            return p;
        }
        p= p->next;
    }
    return NULL;


void deletemalasi(adrmute parent, string namamalte){
        if(parent->firstmalasi == NULL)return;

        adrmalasi cur = parent->firstmalasi;
        adrmalasi prev = NULL;

        while(cur != NULL && cur->child->namamalte != namamalte){
            prev = cur;
            cur =cur->next;
        }

        if(cur == NULL)return;

        if(prev == NULL){
            parent->firstmalasi = cur->next;
        } else {
            prev->next = cur->next;
        }

        delete cur;
```

```
void showChildrenOfParent(adrRute parent){
    //g rute tdaik nemu
    if(parent == NULL){
        cout << "rute tidak ditemukan\n";
        return;
    }

    cout << "rute " << parent->namaRute << " memiliki halte:\n";

    adrRelasi p = parent->firstRelasi;


    //g rute tdk punya relasi
    if(p == NULL){
        cout << "[tidak ada halte]\n";
        return;
    }

    while(p != NULL){
        cout << "- " << p->child->namaHalte << endl;
        p = p->next;
    }


void showAllParentWithChild(listRute lr){
    adrRute r = lr.first;
    while(r != NULL){
        showChildrenOfParent(r);
        r = r->next;
    }


void showParentOfChild(listRute lr, string namaHalte){
    adrRute r = lr.first;
    bool found = false;

    while(r != NULL){
        if(findRelasi(r, namaHalte) != NULL){
            cout <<"rute: " << r->namaRute << endl;
            found = true;
        }
        r = r->next;
    }

    if(!found){
        cout << "Halte tidak memiliki rute(parent)\n";
    }



/coun
nt countRelasiParent(adrRute parent){
    int count = 0;
    adrRelasi p = parent->firstRelasi;
    while(p != NULL){
        count++;
        p = p->next;
    }

    return count;
```

```
131
132
133 int countRelasiChild(ListRute Lr, string namaHalte){
134     int count = 0;
135     adrRute r = Lr.first;
136
137
138     while(r != NULL){
139         if(findRelasi(r, namaHalte) != NULL){
140             count++;
141         }
142         r = r->next;
143     }
144     return count;
145 }
146
147 int countChildNoRelasi(ListHalte Lh, ListRute Lr){
148     int count = 0;
149     adrHalte h = Lh.first;
150
151     while(h != NULL){
152         if(countRelasiChild(Lr, h->namaHalte)== 0){
153             count++;
154         }
155         h = h->next;
156     }
157     return count;
158
159
160 }
161
162 void editRelasi(adrRute parent,ListHalte Lh, string oldHalte, string newHalte){
163     if(parent == NULL)return;
164
165     //cekrelasilama
166     adrRelasi R = findRelasi(parent, oldHalte);
167     if( R == NULL){
168         cout <<" Relasi lama tidak di temukan \n";
169         return;
170     }
171
172     //carihaltebaru
173     adrHalte hNew = findHalte(Lh, newHalte);
174     if(hNew == NULL){
175         cout << "Halte baru tidak ditemukan\n";
176         return;
177     }
178
179     //cekdup
180     if(findRelasi(parent, newHalte) != NULL){
181         cout << "Halte sudah ada di Rute ini\n";
182         return;
183     }
184
185     //gantiR
186     R->child = hNew;
187
188     cout << "Relasi berhasil diubah\n";
189
190 }
191
```

```cpp
void showAllData(ListRute Lr, ListHalte Lh){

    cout << "------ Data Rute & Halte ------\n\n";

    // 1. Parent + Child
    if(Lr.first == NULL){
        cout << "[TIDAK ADA RUTE]\n\n";
    } else {
        adrRute r = Lr.first;
        while(r != NULL){
            cout << "Rute: " << r->namaRute << endl;

            if(r->firstRelasi == NULL){
                cout << "  [TIDAK MEMILIKI HALTE]\n";
            } else {
                adrRelasi rel = r->firstRelasi;
                while(rel != NULL){
                    if(rel->child != NULL){
                        cout << "  - " << rel->child->namaHalte << endl;
                    } else {
                        cout << "  - [RELASI RUSAK]\n";
                    }
                    rel = rel->next;
                }
            }

            cout << endl;
            r = r->next;
        }
    }

    // 2. Child tanpa Parent
    cout << "Halte tanpa Rute:\n";

    if(Lh.first == NULL){
        cout << "[TIDAK ADA HALTE]\n";
    } else {
        adrHalte h = Lh.first;
        bool ada = false;

        while(h != NULL){
            if(countRelasiChild(Lr, h->namaHalte) == 0){
                cout << "- " << h->namaHalte << endl;
                ada = true;
            }
            h = h->next;
        }

        if(!ada){
            cout << "[SEMUA HALTE SUDAH TERHUBUNG]\n";
        }
    }

    cout << "\n--------------------------------\n";
}


// tambahan

void deleteRelasiByHalte(ListRute &Lr, string namaHalte){
    adrRute r = Lr.first;
    while(r != NULL){
        deleteRelasi(r, namaHalte);
        r = r->next;

    }

}
```