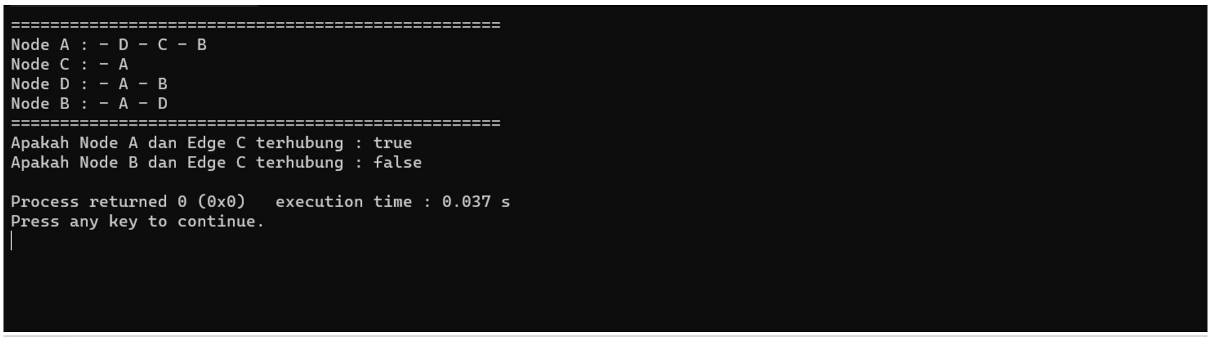
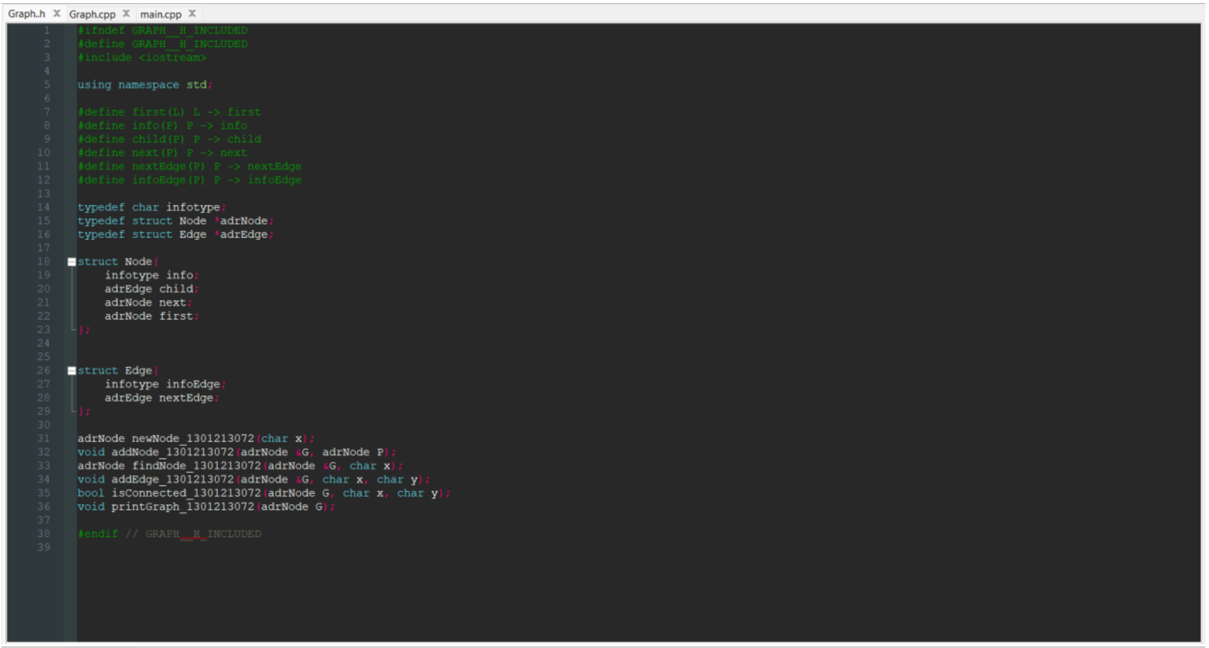
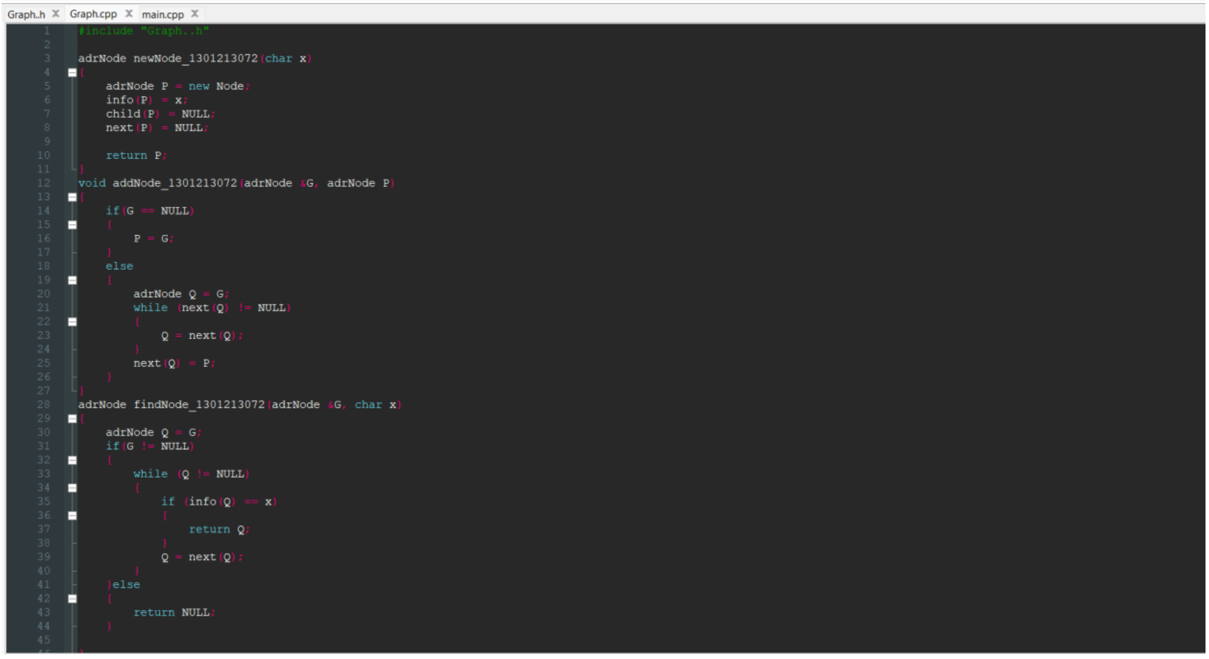


```
==================================================
Node A : - D - C - B
Node C : - A
Node D : - A - B
Node B : - A - D
==================================================
Apakah Node A dan Edge C terhubung : true
Apakah Node B dan Edge C terhubung : false

Process returned 0 (0x0)   execution time : 0.037 s
Press any key to continue.
```



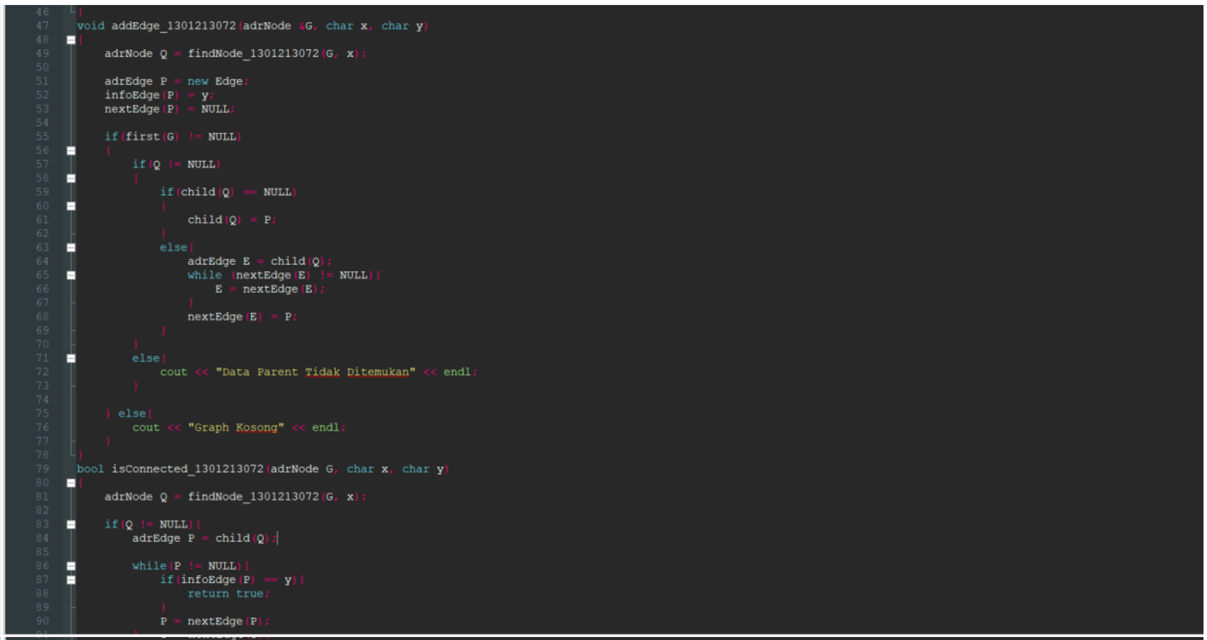
Graph.h   Graph.cpp   main.cpp

```cpp
#ifndef GRAPH__H_INCLUDED
#define GRAPH__H_INCLUDED
#include <iostream>

using namespace std;

#define first(L) L -> first
#define info(P) P -> info
#define child(P) P -> child
#define next(P) P -> next
#define nextEdge(P) P -> nextEdge
#define infoEdge(P) P -> infoEdge

typedef char infotype;
typedef struct Node *adrNode;
typedef struct Edge *adrEdge;

struct Node{
    infotype info;
    adrEdge child;
    adrNode next;
    adrNode first;
};


struct Edge{
    infotype infoEdge;
    adrEdge nextEdge;
};

adrNode newNode_1301213072(char x);
void addNode_1301213072(adrNode &G, adrNode P);
adrNode findNode_1301213072(adrNode &G, char x);
void addEdge_1301213072(adrNode &G, char x, char y);
bool isConnected_1301213072(adrNode G, char x, char y);
void printGraph_1301213072(adrNode G);

#endif // GRAPH__H_INCLUDED

```

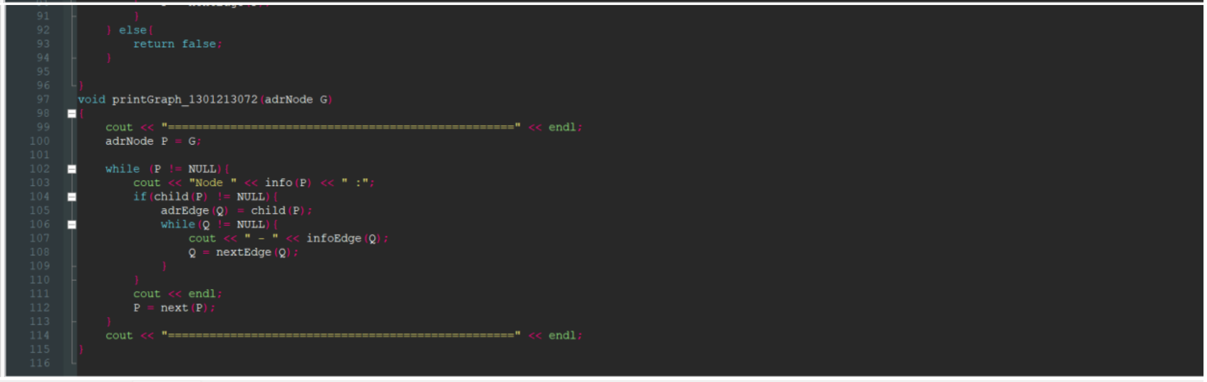


Graph.h   Graph.cpp   main.cpp

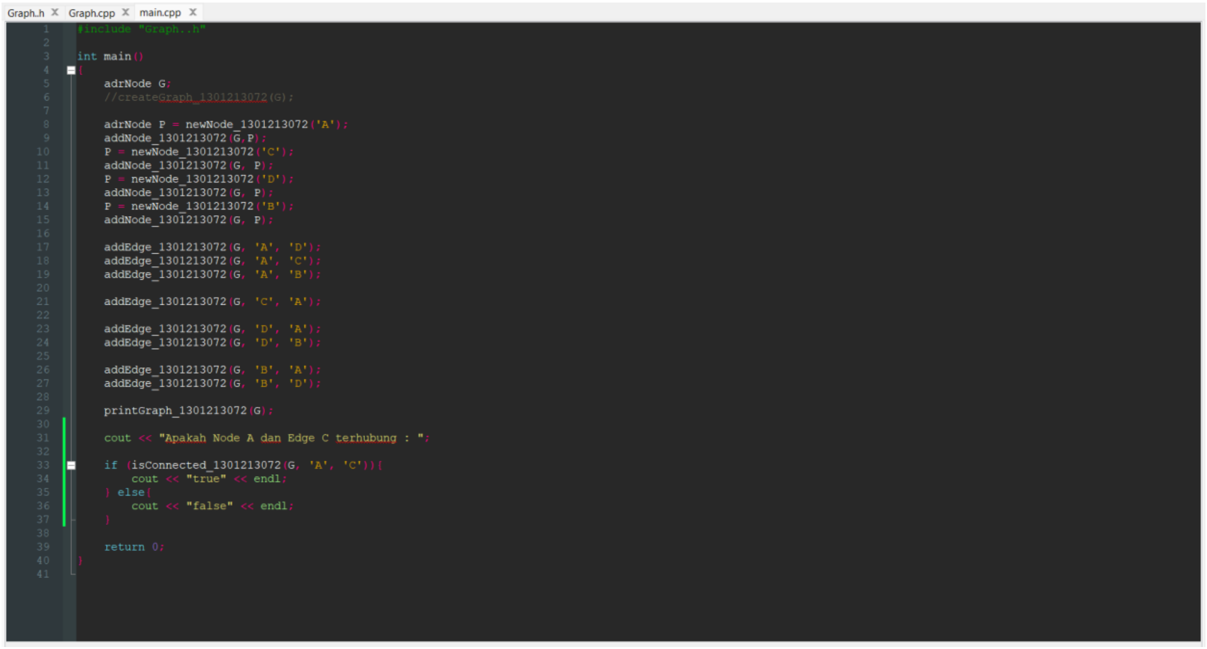
```cpp
#include "Graph..h"

adrNode newNode_1301213072(char x)
{
    adrNode P = new Node;
    info(P) = x;
    child(P) = NULL;
    next(P) = NULL;

    return P;
}
void addNode_1301213072(adrNode &G, adrNode P)
{
    if(G == NULL)
    {
        P = G;
    }
    else
    {
        adrNode Q = G;
        while (next(Q) != NULL)
        {
            Q = next(Q);
        }
        next(Q) = P;
    }
}
adrNode findNode_1301213072(adrNode &G, char x)
{
    adrNode Q = G;
    if(G != NULL)
    {
        while (Q != NULL)
        {
            if (info(Q) == x)
            {
                return Q;
            }
            Q = next(Q);
        }
    }else
    {
        return NULL;
    }

```



```cpp
}
void addEdge_1301213072(adrNode &G, char x, char y)
{
    adrNode Q = findNode_1301213072(G, x);

    adrEdge P = new Edge;
    infoEdge(P) = y;
    nextEdge(P) = NULL;

    if(first(G) != NULL)
    {
        if(Q != NULL)
        {
            if(child(Q) == NULL)
            {
                child(Q) = P;
            }
            else{
                adrEdge E = child(Q);
                while (nextEdge(E) != NULL){
                    E = nextEdge(E);
                }
                nextEdge(E) = P;
            }
        }
        else{
            cout << "Data Parent Tidak Ditemukan" << endl;
        }

    } else{
        cout << "Graph Kosong" << endl;
    }
}
bool isConnected_1301213072(adrNode G, char x, char y)
{
    adrNode Q = findNode_1301213072(G, x);

    if(Q != NULL){
        adrEdge P = child(Q);

        while(P != NULL){
            if(infoEdge(P) == y){
                return true;
            }
            P = nextEdge(P);
```



```cpp
        }
    } else{
        return false;
    }

}
void printGraph_1301213072(adrNode G)
{
    cout << "==================================================" << endl;
    adrNode P = G;

    while (P != NULL){
        cout << "Node " << info(P) << " :";
        if(child(P) != NULL){
            adrEdge(Q) = child(P);
            while(Q != NULL){
                cout << " - " << infoEdge(Q);
                Q = nextEdge(Q);
            }
        }
        cout << endl;
        P = next(P);
    }
    cout << "==================================================" << endl;
}
```



Graph..h Graph.cpp main.cpp

```cpp
#include "Graph..h"

int main()
{
    adrNode G;
    //createGraph_1301213072(G);

    adrNode P = newNode_1301213072('A');
    addNode_1301213072(G,P);
    P = newNode_1301213072('C');
    addNode_1301213072(G, P);
    P = newNode_1301213072('D');
    addNode_1301213072(G, P);
    P = newNode_1301213072('B');
    addNode_1301213072(G, P);

    addEdge_1301213072(G, 'A', 'D');
    addEdge_1301213072(G, 'A', 'C');
    addEdge_1301213072(G, 'A', 'B');

    addEdge_1301213072(G, 'C', 'A');

    addEdge_1301213072(G, 'D', 'A');
    addEdge_1301213072(G, 'D', 'B');

    addEdge_1301213072(G, 'B', 'A');
    addEdge_1301213072(G, 'B', 'D');

    printGraph_1301213072(G);

    cout << "Apakah Node A dan Edge C terhubung : ";

    if (isConnected_1301213072(G, 'A', 'C')){
        cout << "true" << endl;
    } else{
        cout << "false" << endl;
    }

    return 0;
}
```