Main.cpp

graph.cpp graph.h main.cpp

```cpp
#include "graph.h"
int main()
{
    graph G;
    adrVertex P;
    initGraph_103032330054(G);
    buildGraph_103032330054(G);
    P = G.firstVertex;
    while (P != NULL){
        cout << endl << P->idVertex;
        P = P->nextVertex;
    }
}

```

Graph.cpp

```cpp
#include "graph.h"
void createVertex_103032330054(char newVertexID, adrVertex &v){
    v->firstEdge = NULL;
    v->nextVertex = NULL;
    v->idVertex = newVertexID;
}
void initGraph_103032330054(graph &G){
    G.firstVertex = NULL;
}
void addVertex_103032330054(graph &G, char newVertexID){
    adrVertex P = new vertex;
    createVertex_103032330054(newVertexID, P);
    if (G.firstVertex == NULL){
        G.firstVertex = P;
    }else{
        P->nextVertex = G.firstVertex;
        G.firstVertex = P;
    }
}
void buildGraph_103032330054(graph &G){
    char input;
    cin >> input;
    while (input >= 'A' && input <= 'Z'){
        addVertex_103032330054(G, input);
        cin >> input;
    }
}

```

Graph.h

graph.cpp graph.h main.cpp

```h
#ifndef GRAPH_H_INCLUDED
#define GRAPH_H_INCLUDED
#include <iostream>
using namespace std;
typedef struct vertex *adrVertex;
typedef struct edge *adrEdge;
struct vertex{
    char idVertex;
    adrVertex nextVertex;
    adrEdge firstEdge;
};
struct edge{
    char destVertexId;
    int weight;
    adrEdge nextEdge;
};
struct graph{
    adrVertex firstVertex;
};
void createVertex_103032330054(char newVertexID, adrVertex &v);
void initGraph_103032330054(graph &G);
void addVertex_103032330054(graph &G, char newVertexID);
void buildGraph_103032330054(graph &G);
#endif // GRAPH_H_INCLUDED
```

Output

```
"D:\Kuliah\Semester 3\STD\Source-Code-Smt-3\Mod
A
B
C
D
E
F
2

F
E
D
C
B
A
Process returned 0 (0x0)   execution time :
Press any key to continue.
```