1. Stuck

#include<iostream>

#include<conio.h>

using namespace std;

struct STACK

{

int data[5];

int atas;

}tumpukan;

STACK tumpuk;

int main ()

{

int pilihan,baru,i;

tumpuk.atas=-1;

do

{

cout<<"1.Push Data\n";

cout<<"2.Pop Data\n";

cout<<"3.Print Data\n";

cout<<"4.End Data\n";

cout<<"\nPilihan = ";

cin>>pilihan;

switch(pilihan)

{

case 1:

{

if(tumpuk.atas==5-1)

{

cout<<"Tumpukan Penuh";

cout<<endl;

getch();

}

else

{

cout<<"Data yang akan di-push = ";cin>>baru;

tumpuk.atas++;

tumpuk.data[tumpuk.atas]=baru;

}

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

break;

}

case 2:

{

if(tumpuk.atas==-1)

{

cout<<"Tumpukan Kosong";

cout<<endl;

getch();

}

else

{

cout<<"Data yang akan di-pop = "<<tumpuk.data[tumpuk.atas];

tumpuk.atas--;

cout<<endl;

getch();

}

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

break;

}

case 3:

{

if(tumpuk.atas==-1){

cout<<"Tumpukan Kosong";

cout<<endl;

getch();

}

else

{

cout<<"\nData pada stack saat ini adalah : \n";

for(int i=tumpuk.atas; i>=0; i--)

{

cout<<"Nilai ruang ke "<<i<<" : "<<tumpuk.data[i]<<endl;

}

cout<<endl;

getch();

}

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

break;

}

default:

cout<<"\nTidak ada dalam pilihan";

cout<<endl;

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

}

}

while(pilihan!=4);

return 0;

}#include<iostream>

#include<conio.h>

using namespace std;

struct STACK

{

int data[5];

int atas;

}tumpukan;

STACK tumpuk;

int main ()

{

int pilihan,baru,i;

tumpuk.atas=-1;

do

{

cout<<"1.Push Data\n";

cout<<"2.Pop Data\n";

cout<<"3.Print Data\n";

cout<<"4.End Data\n";

cout<<"\nPilihan = ";

cin>>pilihan;

switch(pilihan)

{

case 1:

{

if(tumpuk.atas==5-1)

{

cout<<"Tumpukan Penuh";

cout<<endl;

getch();

}

else

{

cout<<"Data yang akan di-push = ";cin>>baru;

tumpuk.atas++;

tumpuk.data[tumpuk.atas]=baru;

}

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

break;

}

case 2:

{

if(tumpuk.atas==-1)

{

cout<<"Tumpukan Kosong";

cout<<endl;

getch();

}

else

{

cout<<"Data yang akan di-pop = "<<tumpuk.data[tumpuk.atas];

tumpuk.atas--;

cout<<endl;

getch();

}

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

break;

}

case 3:

{

if(tumpuk.atas==-1){

cout<<"Tumpukan Kosong";

cout<<endl;

getch();

}

else

{

cout<<"\nData pada stack saat ini adalah : \n";

for(int i=tumpuk.atas; i>=0; i--)

{

cout<<"Nilai ruang ke "<<i<<" : "<<tumpuk.data[i]<<endl;

}

cout<<endl;

getch();

}

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

break;

}

default:

cout<<"\nTidak ada dalam pilihan";

cout<<endl;

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

}

}

while(pilihan!=4);

return 0;

}

1. Queue
2. #include <iostream>
3. #define n 10
4. using namespace std;
5. struct mahasiswa{
6. string nama;
7. int nim;
8. }Mhs[n];
9. struct queue{
10. int awal, akhir;
11. }Queue;
12. void deQueue();
13. void menu();
14. void inQueue();
15. void tampil();
16. void init(){
17. Queue.awal = -1;
18. Queue.akhir = -1;
19. cout << "Data berhasil di reset" << endl << endl;
20. menu();
21. }
22. bool isFull(){
23. if(Queue.akhir == n - 1){
24. return true;
25. } else {
26. return false;
27. }
28. }
29. bool isEmpty(){
30. if(Queue.akhir == -1){
31. return true;
32. }else {
33. return false;
34. }
35. }
36. void inQueue(){
37. if(!isFull()){
38. string nama;
39. int nim;
40. cout << "Masukkan nama mahasiswa:";
41. cin >> nama;
42. cout << "Masukkan NIM mahasiswa:";
43. cin >> nim;
44. Mhs[Queue.akhir].nama = nama;
45. Mhs[Queue.akhir].nim = nim;
46. ++Queue.akhir;
47. cout << endl;
48. menu();
49. }else {
50. cout << "Data penuh";
51. menu();
52. }
53. }
54. void menu(){
55. int pilih;
56. cout << "1. masukkan data\n";
57. cout << "2. Hapus satu data\n";
58. cout << "3. Reset data\n";
59. cout << "4. tampil data\n";
60. cout << "Masukkan pilihan anda:";
61. cin >> pilih;
62. cout << endl;
63. if(pilih == 1){
64. inQueue();
65. }else if(pilih == 2){
66. deQueue();
67. } else if(pilih == 3){
68. init();
69. }else if(pilih == 4){
70. tampil();
71. }else {
72. menu();
73. }
74. }
75. void deQueue(){
76. if(!isEmpty()){
77. for(int i = Queue.awal; i < Queue.akhir; i++){
78. Mhs[i].nama = Mhs[i+1].nama;
79. Mhs[i].nim = Mhs[i+1].nim;
80. }Queue.akhir--;
81. cout << "Data berhasil dihapus" << endl;
82. cout << endl;
83. menu();
84. }else{
85. cout << "antrian kosong" << endl;
86. }
87. }
88. void tampil(){
89. if(!isEmpty()){
90. for(int i = 0; i < Queue.akhir; i++){
91. cout << "Nama Mahasiswa: " << Mhs[i].nama << endl;
92. cout << "NIM : " << Mhs[i].nim << endl;
93. cout << endl;
94. }
95. }else {
96. cout << "data kosong";
97. cout << endl << endl;
98. }
99. menu();
100. }
101. int main(){
102. menu();
103. return 0;
104. }

3. #include <stdio.h>

#include <conio.h>

#include <stdlib.h>

//pendeklarasian struct sebuah tree awal

struct Node{

      int data;

      Node \*kiri;

      Node \*kanan;

};

//fungsi untuk menambahkan node baru

void tambah(Node \*\*root, int databaru)

{

      //jika root masih kosong

      if((\*root) == NULL)

      {

            //pembuatan node baru

            Node \*baru;

            //pengalokasian memori dari node yang telah dibuat

            baru = new Node;

            //inisialisasi awal node yang baru dibuat

            baru->data = databaru;

            baru->kiri = NULL;

            baru->kanan = NULL;

            (\*root) = baru;

            (\*root)->kiri = NULL;

            (\*root)->kanan = NULL;

            printf("Data bertambah!");

      }

     //jika data yang akan dimasukkan lebih kecil daripada elemen root, maka akan diletakkan di node sebelah kiri.

      else if(databaru<(\*root)->data)

            tambah(&(\*root)->kiri, databaru);

     //jika data yang akan dimasukkan lebih besar daripada elemen root, maka akan diletakkan di node sebelah kanan

      else if(databaru>(\*root)->data)

            tambah(&(\*root)->kanan, databaru);

     //jika saat dicek data yang akan dimasukkan memiliki nilai yang sama dengan data pada root

      else if(databaru == (\*root)->data)

            printf("Data sudah ada!");

}

//fungsi yang digunakan untuk mencetak tree secara preOrder

void preOrder(Node \*root)

{

      if(root != NULL){

            printf("%d ", root->data);

            preOrder(root->kiri);

            preOrder(root->kanan);

      }

}

//fungsi yang digunakan untuk mencetak tree secara inOrder

void inOrder(Node \*root)

{

      if(root != NULL){

            inOrder(root->kiri);

            printf("%d ", root->data);

            inOrder(root->kanan);

      }

}

//fungsi yang digunakan untuk mencetak tree secara postOrder

void postOrder(Node \*root)

{

      if(root != NULL){

            postOrder(root->kiri);

            postOrder(root->kanan);

            printf("%d ", root->data);

      }

}

//fungsi utama

int main()

{

      //deklarasikan variabel

      int pil, data;// c;

      Node \*pohon; //\*t;

      pohon = NULL; //inisialisasi node pohon

      //perulangan do-while

      do

      {

            system("cls"); //bersihkan layar

            printf("\t#PROGRAM TREE C++#");

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

            printf("\nMENU");

            printf("\n----\n");

            printf("1. Tambah\n");

            printf("2. Lihat pre-order\n");

            printf("3. Lihat in-order\n");

            printf("4. Lihat post-order\n");

            printf("5. Exit\n");

            printf("Pilihan : ");

            scanf("%d", &pil);

            switch(pil)

            {

            //jika pil bernilai 1

            case 1 :

                  printf("\nINPUT : ");

                  printf("\n-------");

                  printf("\nData baru : ");

                  scanf("%d", &data);

                  //panggil fungsi untuk menambah node yang berisi data pada tree

                  tambah(&pohon, data);

                  break;

            //jika pil bernilai 2

            case 2 :

                  printf("\nOUTPUT PRE ORDER : ");

                  printf("\n------------------\n");

                  if(pohon!=NULL)

                       //panggil fungsi untuk mencetak data secara preOrder

                        preOrder(pohon);

                  else

                        printf("Masih kosong!");

                  break;

            //jika pil bernilai 3

            case 3 :

                  printf("\nOUTPUT IN ORDER : ");

                  printf("\n------------------\n");

                  if(pohon!=NULL)

                       //panggil fungsi untuk mencetak data secara inOrder

                        inOrder(pohon);

                  else

                        printf("Masih kosong!");

                  break;

            //jika pil bernilai 4

            case 4 :

                  printf("\nOUTPUT POST ORDER : ");

                  printf("\n------------------\n");

                  if(pohon!=NULL)

                       //panggil fungsi untuk mencetak data secara postOrder

                        postOrder(pohon);

                  else

                        printf("Masih kosong!");

                  break;

            }

            \_getch();

      }while(pil != 5); //akan diulang jika input tidak samadengan 5

      return EXIT\_FAILURE;