Giullio Emmanuel da Cruz Di Gerolamo

RA: 790965

Frequência F2

Arquivo Pilha.cpp

```
#include <iostream>


using namespace std;


class pilha {

   private:
      static const int tam = 5; // Tamanho máximo da pilha
      int topo; // variável que controla qual o topo atual da pilha
      int itens[tam]; // Pilha a ser inseridos dados

   public:

      /*
       * Inicializa as variáveis
       */
      pilha ()
      {
         topo = -1;
      }

      /*
       * Empilha um item na Pilha
       */
      void Push (int item)
      {
         if(topo >= tam - 1) {
            cout << "Stack overflow" << endl;
         } else {
            itens[++topo] = item;
         }
      }

      /*
       * Desempilha o último item da lista
```

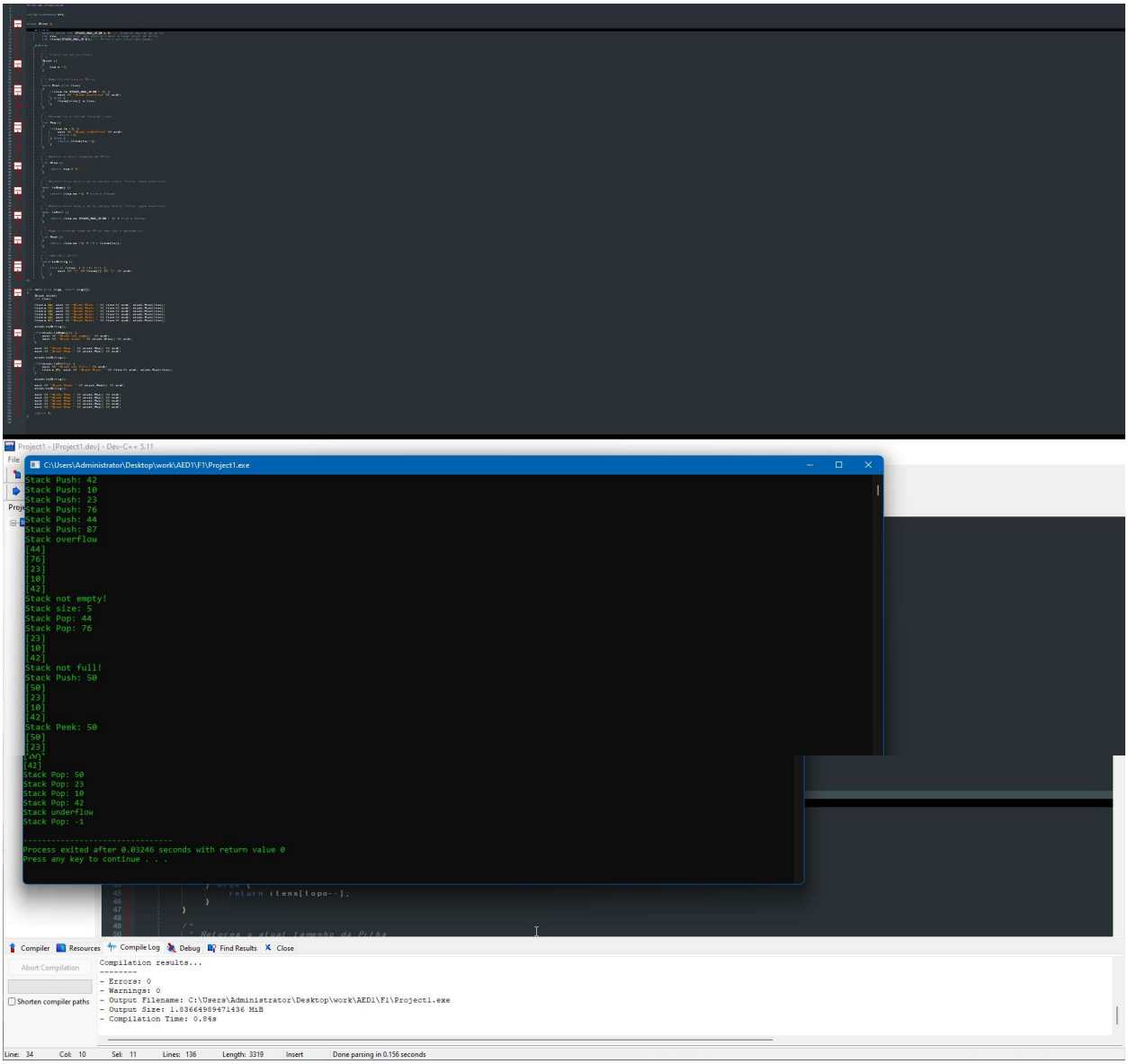
```
     */
    int Pop ()
    {
       if(topo <= -1) {
          cout << "Stack underflow" << endl;
          return -1;
       } else {
          return itens[topo--];
       }
    }

    /*
     * Retorna o atual tamanho da Pilha
     */
    int Tamanho ()
    {
       return topo + 1;
    }

    /*
     * Retorna true caso a pilha esteja vazia. false, caso contrário
     */
    bool Vazio ()
    {
       return (topo == -1) ? true : false;
    }

    /*
     * Retorna true caso a pilha esteja cheia. false, caso contrário
     */
    bool Cheio ()
    {
       return (topo == tam - 1) ? true : false;
    }

    /*
     * Pega o item do topo da Pilha mas não o desempilha
     */
    int Peek ()
    {
       return (topo == -1) ? -1 : itens[topo];
    }

    /*
     * Imprime a pilha
     */
    void toString ()
    {
       for(int i=topo; i > -1; i--) {
          cout << "[" << itens[i] << "]" << endl;
       }
    }
};
```

```
int main (int argc, char* argv[])
{
   pilha pilha;
   int item;

   item = 42; cout << "Stack Push: " << item << endl; pilha.Push(item);
   item = 10; cout << "Stack Push: " << item << endl; pilha.Push(item);
   item = 23; cout << "Stack Push: " << item << endl; pilha.Push(item);
   item = 76; cout << "Stack Push: " << item << endl; pilha.Push(item);
   item = 44; cout << "Stack Push: " << item << endl; pilha.Push(item);
   item = 87; cout << "Stack Push: " << item << endl; pilha.Push(item);

   pilha.toString();

   if(!pilha.Vazio()) {
      cout << "pilha nao esta vazia!" << endl;
      cout << "tamanho da pilha: " << pilha.Tamanho() << endl;
   }

   cout << "Stack Pop: " << pilha.Pop() << endl;
   cout << "Stack Pop: " << pilha.Pop() << endl;

   pilha.toString();

   if(!pilha.Cheio()) {
      cout << "Pilha nao esta cheia!" << endl;
      item = 50; cout << "Stack Push: " << item << endl; pilha.Push(item);
   }

   pilha.toString();

   cout << "topo da pilha: " << pilha.Peek() << endl;
   pilha.toString();

   cout << "Stack Pop: " << pilha.Pop() << endl;
   cout << "Stack Pop: " << pilha.Pop() << endl;
   cout << "Stack Pop: " << pilha.Pop() << endl;
   cout << "Stack Pop: " << pilha.Pop() << endl;
   cout << "Stack Pop: " << pilha.Pop() << endl;

   return 0;
}
```

Uma possível aplicação para uma Pilha

Uma comum aplicação de uma pilha no Sistema é o desfazer/refazer de editors de texto. Estes por sua vez armazenam os dados que foram apagados anteriormente em uma pilha para caso necessario refazer.

Prints da execução



Essa foi minha 2 tentativa de implementação, depois de não conseguir implementar baseado em struct.