

|  |
| --- |
| Binary Tree (배열 구현방식) |
| **과제 5장 보고서(1번 - 배열 구현방식)** |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **제 출 일** | **2014. 09. 11** |  | **전 공** | **Business & CSE** |
| **과 목** | **자료구조론** |  | **학 번** | **20101215** |
| **담당교수** | **이 현 아** |  | **이 름** | **정 원 영** |

**1. Binary Tree(배열)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Complete Binary Tree의 원리**   |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | |  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |   **Leftorder의 원리:** 2 \* index  **Rightorder의 원리:** 2\* index + 1  **Index = 0 일 때,**  - 2 \* 0 = 0, …..  - (2 \* 0) + 1 = 1, (2 \* 1) + 1 = 3, ….  **Index = 1일 때,**  - 2 \* 1 = 2, 2 \* 2 = 4, …… , 이런식으로 진행  - (2 \* 1) + 1 = 3, (2 \* 3) + 1 = 7, …… , ……  **증명)**   |  |  |  | | --- | --- | --- | | **Preorder** | **Inorder** | **Postorder** | | template <class T>  void BinaryTree<T>::Preorder(int index){  if (pArr[index] == 0 || index > GetMaxSize())  return;  cout << pArr[index] << " ";  Preorder(2 \* index);  Preorder(2 \* index + 1);  } | template <class T>  void BinaryTree<T>::Inorder(int index){  if (pArr[index] == 0 || index > GetMaxSize())  return;  Inorder(index \* 2);  cout << pArr[index] << " ";  Inorder(index \* 2 + 1);  } | template <class T>  void BinaryTree<T>::Postorder(int index){  if (pArr[index] == 0 || index > GetMaxSize())  return;  Postorder(index \* 2);  Postorder(index \* 2 + 1);  cout << pArr[index] << " ";  } | | **Levelorder** | | | | template <class T>  void BinaryTree<T>::Levelorder(int index){  if (pArr[index] == 0 || index > GetMaxSize())  return;  cout << pArr[index] << " ";  Levelorder(index + 1);  } | | |   **자료구조론 교제에 있는 원리를 적용해본 결과** |

**2. 코드 결과**

|  |
| --- |
| /\*  금오공과대학교  과제: 방학(Binary Tree) 과제 - 1번(배열 구현)  학과: Business & CSE  학번: 20101215  이름: 정원영  \*/  #include <iostream>  #include <string>  #include "io.h"  #include "binarytree.h"  using namespace std;  int main()  {  int Count = 0;  int UserSize; // 노드 사이즈 - 사용자가 지정  BinaryTree<int> Tree; // 트리 생성  string UserData;  cout << "노드 개수를 입력하세요 : ";  cin >> UserSize;  // 1. 데이터 입력  while (Count != UserSize)  {  string TmpData;  std::getline(cin, TmpData);    int CurCount = Tree.GetCount(TmpData);  // (CurrentLevel + Count)로 범위 파악 (UserSize 비교)  if ((Count + CurCount) > UserSize)  cout << "노드 갯수 초과" << endl;  else  {  TmpData.append("\n");  UserData.append(TmpData);  Count += CurCount;  }  } // End of while    // 2. Binary Tree  Tree.SetMaxSize(UserSize);  Tree.SetData(UserData);  // Traversal 처리  cout << "preorder : ";  Tree.Preorder(1);  cout << endl;  cout << "inorder : ";  Tree.Inorder(1);  cout << endl;  cout << "postorder : ";  Tree.Postorder(1);  cout << endl;  cout << "level order : ";  Tree.Levelorder(1);  cout << endl;  // 소멸자 처리  Tree.DestoryTree();  }  // String To Char(자료형 변환)  const char StrToCh(std::string Data){  return Data.at(0);  } |

Figure 1) Main.cpp

|  |
| --- |
| /\*  금오공과대학교  과제: 방학(Binary Tree) 과제 - 1번(배열 구현)  학과: Business & CSE  학번: 20101215  이름: 정원영  \*/  #ifndef \_BINARYTREE\_H\_  #define \_BINARYTREE\_H\_  #include <iostream>  #include <string>  template <class T>  class BinaryTree{  public:  void Preorder(int index);  void Inorder(int index);  void Postorder(int index);  void Levelorder(int index);  void SetData(std::string Data);  int GetCount(std::string Data);  int GetMaxSize();  void SetMaxSize(int n);  void DestoryTree();  private:  int MAX\_SIZE;  T \*pArr;  };  #include "binarytree\_detail.h"  #endif |

Figure 2) binarytree.h

|  |
| --- |
| /\*  금오공과대학교  과제: 방학(Binary Tree) 과제 - 1번(배열 구현)  학과: Business & CSE  학번: 20101215  이름: 정원영  \*/  #ifndef \_BINARYTREE\_DETAIL\_H\_  #define \_BINARYTREE\_DETAIL\_H\_  #include "binarytree.h"  const char StrToCh(std::string Data);  template <class T>  void BinaryTree<T>::Preorder(int index){  if (pArr[index] == 0 || index > GetMaxSize())  return;  cout << pArr[index] << " ";  Preorder(2 \* index);  Preorder(2 \* index + 1);  }  template <class T>  void BinaryTree<T>::Inorder(int index){  if (pArr[index] == 0 || index > GetMaxSize())  return;  Inorder(index \* 2);  cout << pArr[index] << " ";  Inorder(index \* 2 + 1);  }  template <class T>  void BinaryTree<T>::Postorder(int index){  if (pArr[index] == 0 || index > GetMaxSize())  return;  Postorder(index \* 2);  Postorder(index \* 2 + 1);  cout << pArr[index] << " ";  }  template <class T>  void BinaryTree<T>::Levelorder(int index){  if (pArr[index] == 0 || index > GetMaxSize())  return;  cout << pArr[index] << " ";  Levelorder(index + 1);  }  // 트리에 실제 데이터 입력 - SetData(string Data);  template <class T>  void BinaryTree<T>::SetData(std::string Data){  int EndIDX = Data.length(); // 문자열의 길이 반환  int Count = 1;  // 공간 초기화  pArr = new T[GetMaxSize() + 1];  for (int i = 0; i < (GetMaxSize() + 1); i++)  pArr[i] = 0;  // 데이터 입력  string TmpData;  for (int StartIDX = 0; StartIDX < EndIDX; StartIDX++)  {  const char chData = StrToCh(Data.substr(StartIDX, 1));  // 숫자 판별  if (isdigit(chData))  TmpData.push\_back(chData);  // 공백 판별  if (chData == ' ')  {  if (!TmpData.empty())  {  pArr[atoi(TmpData.c\_str())] = atoi(TmpData.c\_str());  Count++;  TmpData.erase();  } // End of if  } // End of if  // 마지막 문자 판별  if ((StartIDX + 1) == EndIDX)  {  if (isdigit(chData))  TmpData.push\_back(chData);  if (!TmpData.empty())  {  pArr[atoi(TmpData.c\_str())] = atoi(TmpData.c\_str());  Count++;  TmpData.erase();  }  } // End of if  } // End of while  }  // 갯수 파악용 - GetCount(string Data);  template <class T>  int BinaryTree<T>::GetCount(std::string Data){  int EndIDX = Data.length(); // 문자열의 길이 반환  int Count = 0;  string TmpData;  for (int StartIDX = 0; StartIDX < EndIDX; StartIDX++)  {  const char chData = StrToCh(Data.substr(StartIDX, 1));  // 숫자 판별  if (isdigit(chData))  TmpData.push\_back(chData);  // 공백 판별  if (chData == ' ')  {  if (!TmpData.empty())  {  Count++;  TmpData.erase();  } // End of if  } // End of if  // 마지막 문자 판별  if ((StartIDX + 1) == EndIDX)  {  if (isdigit(chData))  TmpData.push\_back(chData);  if (!TmpData.empty())  {  Count++;  TmpData.erase();  }  } // End of if  } // End of while  return Count;  }  template <class T>  void BinaryTree<T>::SetMaxSize(int n){  MAX\_SIZE = n;  }  template <class T>  int BinaryTree<T>::GetMaxSize(){  return MAX\_SIZE;  }  template <class T>  void BinaryTree<T>::DestoryTree(){  delete pArr;  }  #endif |

Figure 3) binarytree\_detail.h