

|  |
| --- |
| Binary Tree (계산식) |
| **과제 5장 보고서(계산식)** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **제 출 일** | **2014. 0** |  | **전 공** | **Business & CSE** |
| **과 목** | **자료구조론** |  | **학 번** | **20101215** |
| **담당교수** |  |  | **이 름** | **정 O O** |

**1. Binary Tree**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **\* 원리 소개**  계산식의 3 + 45 – 78 \* 5 – 2^3\*(5-2)  3장의 과제에서 살펴본 것처럼, Postfix로 전환해야 한다.  Postfix로 전환하면,  **3 45 + 78 ~ 5 \* - 2 3 ^ 5 2 - \* -**  **\* 트리 구축에 관한 계획 수립**   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1 | | | 2 | | | 3 | | | | X | **Y** | **Result** | **X** | **Y** | **Result** | **X** | **Y** | **Result** | | 3 |  |  | 78 |  |  | -78 | 5 |  | |  | 45 | + |  | ~ |  |  | x |  | |  |  | 48 |  |  | -78 |  |  | -390 | | 4 | | | **5** | | | **6** | | | | X | **Y** | **Result** | **X** | **Y** | **Result** | **X** | **Y** | **Result** | | 48 | -390 |  | 2 | 3 |  | 3 | 8 |  | |  | - | 438 |  | ^ | 8 |  | x | 24 | |  |  |  |  |  |  |  |  |  | | 7 | | | | X | **Y** | **Result** | | 24 | - 438 |  | |  | - | 414 | |  |  |  |   **Postfix로 된 표현을 연산했을 때 각 변수에 담기는 모습들 관찰 결과**  **1단계)**  **pStack->Push**  **2단계)**  **pStack->Push**  **3단계) y == -78**    **4단계) xNode->pop(), yNode->pop()**  **pStack->Push**  **5단계)**  **pStack->Push**  **6단계)**  **pStack->Push**  **7단계)**  **pStack->Push**  **8단계)**  **xNode->pop(), yNode->pop()**  **pStack->Push**  **\* LRV(Traversal)**   |  |  | | --- | --- | | **정리) Case1)**  ‘~’ | **Case 2)**  입력: ‘^’  연산: push | | **Case 3)**  기존 top이 ‘^’일 때  Push 모드 | |  |  | | --- | --- | | **Case 4)** | | | **Left (외부)**  **Right (외부)** |  | | **스택(내부) Left**  **스택(내부) Right** |  | | |

**3. 코드 결과**

|  |
| --- |
| /\*  OOOOOOOOOOOOO교  과제: 방학(Binary Tree) 과제5 - 2번(계산식)  학과: Business & CSE  학번: 20101215  이름: 정OO  \*/  #include <iostream>  #include <istream>  #include <string>  #include <math.h>  #include "tree.h"  #include "stack.h"  using namespace std;  int eval(char \*post);  void postfix(char \*dst, char \*src);  int is\_legal(char \*postfix);  int precedence(int op);  bool is\_operator(int op);  char\* blankToChar(char\* dest);  char\* createBlankChar(int size);  int is\_quit(char\* dest);  int main()  {  int result;  while (1)  {  char \*exp = createBlankChar(256);  char Infix[256];  cout << "계산식을 입력하세요 (종료는 quit) : ";  cin.clear();  cin.getline(Infix, 256);  if (is\_quit(Infix))  break;  postfix(exp, blankToChar(Infix));  if (!is\_legal(exp))  {  cout << "정상적인 표현이 아닙니다." << endl;  continue;  }  else  {  cout << "--- 후위 계산식은 : " << exp << endl;  result = eval(exp);  cout << "---- 결과는 " << result << endl;  }  }  return 0;  }  char\* createBlankChar(int size){  char\* exp = new char[size];  for (int i = 0; i < size; i++)  exp[i] = NULL;  return exp;  }  int is\_quit(char\* dest){  int idx = 0;  const int q = 0;  const int u = 1;  const int i = 2;  const int t = 3;  const int quit = 4;  bool check = true;  while (\*dest){  switch (idx)  {  case q:  if (\*dest != 'q')  check = false;  break;  case u:  if (\*dest != 'u')  check = false;  break;  case i:  if (\*dest != 'i')  check = false;  break;  case t:  if (\*dest != 't')  check = false;  break;  default:  if (idx > t)  check = false;  break;  }  idx++;  \*dest++;  }  if (idx != quit)  check = false;  return check;  }  char\* blankToChar(char\* dest){  char\* pDest = new char[256];  int i = 0;  while (i < 256)  {  pDest[i++] = ' ';  }  i = 0;  while (\*dest)  {  if (\*dest != ' ')  pDest[i++] = \*dest;  \*dest++;  }  return pDest;  }  bool is\_operator(int op)  {  bool result = false;  if (op == '+' || op == '-')  result = true;  if (op == '\*' || op == '^')  result = true;  if (op == '/' || op == '^')  result = true;  if (op == '%')  result = true;  return result;  }  int precedence(int op)  {  if (op == '(') return 0;  if (op == '+' || op == '-') return 1;  if (op == '\*' || op == '/') return 2;  if (op == '^') return 3;  else return 4;  }  int is\_legal(char \*postfix)  {  int result = 0;  if (\*postfix == NULL)  return 0;  while (\*postfix)  {  while (\*postfix == ' ')  postfix++;  if (is\_operator(\*postfix))  result--;  else  {  if (\*postfix != '~')  result++;  while (\*postfix != ' ')  postfix++;  } // End of if  if (result < 1) break;  postfix++;  }  return (result == 1);  }  void postfix(char \*dst, char \*src)  {  Stack<char> \*PostStack = new Stack<char>();  Stack<char> \*MinusStack = new Stack<char>();  while (\*src)  {  if (\*src == '(')  {  PostStack->push(\*src);  src++;  }  else if (\*src == ')')  {  while (PostStack->get\_stack\_top() != '(')  {  \*dst++ = PostStack->pop();  \*dst++ = ' ';  }  PostStack->pop();  src++;  }  else if (is\_operator(\*src))  {  while (!PostStack->is\_stack\_empty() && precedence(PostStack->get\_stack\_top()) >= precedence(\*src))  {  \*dst++ = PostStack->pop();  \*dst++ = ' ';  }  if (\*src == '-' && src[1] == '-')  MinusStack->push('~');  else  PostStack->push(\*src);  src++;  }  else if (\*src >= '0' && \*src <= '9')  {  do  {  \*dst++ = \*src++;  } while (\*src >= '0' && \*src <= '9');  if (!MinusStack->is\_stack\_empty())  {  \*dst++ = ' ';  \*dst++ = MinusStack->pop();  }  \*dst++ = ' ';  }  else  src++;  }  while (!PostStack->is\_stack\_empty())  {  \*dst++ = PostStack->pop();  \*dst++ = ' ';  }  dst--;  \*dst = 0;  }  int eval(char \*post)  {  int left, right, result;  Tree\* pTree = new Tree();  Stack<int> \*PostStack = new Stack<int>();  while (\*post)  {  if (\*post >= '0' && \*post <= '9')  {  result = 0;  do  {  result = result \* 10 + \*post - '0';  post++;  } while (\*post >= '0' && \*post <= '9');  PostStack->push(result);  }  else if (\*post == '+')  {  right = PostStack->pop();  left = PostStack->pop();  result = right + left;  PostStack->push(result);  pTree->Make(left, right, result, '+');  post++;  }  else if (\*post == '\*')  {  right = PostStack->pop();  left = PostStack->pop();  result = right \* left;  PostStack->push(result);  pTree->Make(left, right, result, '\*');  post++;  }  else if (\*post == '-')  {  right = PostStack->pop();  left = PostStack->pop();  result = left - right;  PostStack->push(result);  pTree->Make(left, right, result, '-');  post++;  }  else if (\*post == '/')  {  right = PostStack->pop();  left = PostStack->pop();  result = left / right;  PostStack->push(result);  pTree->Make(left, right, result, '/');  }  else if (\*post == '^')  {  right = PostStack->pop();  left = PostStack->pop();  result = (int)pow((double)left, (double)right);  PostStack->push(result);  pTree->Make(left, right, result, '^');  post++;  }  else if (\*post == '%')  {  right = PostStack->pop();  left = PostStack->pop();  result = left % right;  PostStack->push(result);  pTree->Make(left, right, result, '%');  post++;  }  else if (\*post == '~')  {  left = PostStack->pop();  right = NULL;  PostStack->push((-1) \* left);  pTree->Make(left, NULL, NULL, '~');  post++;  }  else  post++;  } // End of if  // 초기 라인스타일  pTree->PrintOut(pTree->GetTree());  result = PostStack->pop();    return result;  } |

Figure 1) Main.cpp

|  |
| --- |
| /\*  OOOOOOOOOOOOO교  과제: 방학(Binary Tree) 과제5 - 2번(계산식)  학과: Business & CSE  학번: 20101215  이름: 정OO  \*/  #ifndef \_TREE\_H\_  #define \_TREE\_H\_  #include <iostream>  #include <string>  using namespace std;  const int ROOT = 1;  const int NODE = 2;  class TreeNode;  class PrintNode;  class pNode{  public:  TreeNode\* Data;  pNode\* prev;  pNode\* next;  };  class pStack{  public:  pStack();  ~pStack();  void push(TreeNode\* \_pNode);  TreeNode\* pop();  int IsEmpty();  int GetCount();  TreeNode\* GetTop();  private:  int Count;  pNode\* Top;  };  class TreeNode{  public:  TreeNode\* LeftNode;  std::string Data;  TreeNode\* RightNode;  };  class PrintNode{  public:  int Height = 0;  std::string Data;  PrintNode\* prev;  PrintNode\* next;  };  class PrintStack{  public:  PrintStack();  ~PrintStack();  void push(std::string \_Data, int \_Height);  std::string pop();  int GetCount();  std::string at(int height);  private:  int Count;  PrintNode\* Top;  };  class Tree{  public:  Tree();  ~Tree();  TreeNode\* Create(TreeNode\* left, TreeNode\* root, TreeNode\* right);  TreeNode\* Create(int number);  TreeNode\* Create(int number, char op);  void DestoryTree(TreeNode\* pNode);  void DestoryNode(TreeNode\* pNode);  void Make(int x, int y, int result, char op);  void PostOrder(TreeNode\* pTreeNode, int level = 0, PrintStack\* pStackNode = NULL);  void PrintOut(TreeNode\* pNode);  int SkewedHeight(int Type, TreeNode\* pNode);  TreeNode\* GetTree();  private:  int resultToInt(std::string D);  std::string GetResult(int number, char op);  pStack\* newStack;  protected:  void SetLine(bool check);  void SetData(int Type, std::string Data);  bool GetLine();  bool printLine;  std::string RootData;  std::string TreeData;  };  #include "tree\_detail.h"  #endif |

**Figure 2) tree.h**

|  |
| --- |
| /\*  OOOOOOOOOOOOO교  과제: 방학(Binary Tree) 과제5 - 2번(계산식)  학과: Business & CSE  학번: 20101215  이름: 정OO  \*/  #ifndef \_TREE\_DETAIL\_H\_  #define \_TREE\_DETAIL\_H\_  #include <iostream>  #include <sstream>  #include <string>  #include <iomanip>  #include "tree.h"  using namespace std;  std::string IntToString(int number);  std::string CharToString(char op);  const char stringToChar(std::string D, int i);  Tree::Tree(){  newStack = new pStack();  }  Tree::~Tree(){  DestoryTree(newStack->pop());  }  void Tree::DestoryTree(TreeNode\* pNode){  if (pNode != NULL)  {  DestoryTree(pNode->LeftNode);  DestoryTree(pNode->RightNode);  DestoryNode(pNode);  }  }  void Tree::DestoryNode(TreeNode\* pNode){  delete pNode;  }  TreeNode\* Tree::GetTree(){  return newStack->GetTop();  }  void Tree::SetLine(bool check){  printLine = check;  }  bool Tree::GetLine(){  return printLine;  }  void Tree::PrintOut(TreeNode\* pNode){  PrintStack\* newData = new PrintStack();  PostOrder(pNode, 0, newData);  cout << endl;  }  void Tree::SetData(int Type, std::string Data){  switch (Type)  {  case ROOT:  RootData = Data;  break;  case NODE:  TreeData.append(Data);  break;  }  }  void Tree::PostOrder(TreeNode\* pTreeNode, int level, PrintStack\* pStackNode){  string Data;  // TreeNode가 아닐 때  if (pTreeNode != NULL)  {  // 라인 바꾸기  if (GetLine() == true)  {  string pData;  int i = 0;  int iStartNum = level;  int iEndNum = pStackNode->GetCount();  if (GetLine() == true){  while (iStartNum < iEndNum)  {  pStackNode->pop();  iStartNum++;  }  }  while (i < pStackNode->GetCount())  {  string pTmpData = pStackNode->at(i);  pData.append(pTmpData);  i++;  }  cout << endl;  i = 0;  while (i <= pData.length())  {  cout << " ";  i++;  }  SetLine(false);  }  if (level == 0)  cout << pTreeNode->Data;  else  cout << " --- " << pTreeNode->Data;  // Node  if (pTreeNode->LeftNode == NULL && pTreeNode->LeftNode == NULL)  {  SetLine(true);  }  else  {  // 루트 위치 설정  if (level == 0)  {  Data.append(pTreeNode->Data);  SetData(ROOT, pTreeNode->Data);  }  // 노드 위치 설정  if (level != 0)  {  if (pTreeNode->Data != "~")  {  Data.append(" --- ");  Data.append(pTreeNode->Data);  }  }  // cout << Data; - Stack 내용  if (Data != ""){  pStackNode->push(Data, level);  }  }  PostOrder(pTreeNode->LeftNode, level + 1, pStackNode);  PostOrder(pTreeNode->RightNode, level + 1, pStackNode);  }  }  std::string Tree::GetResult(int number, char op){  std::string pData;  if (op == '~'){  pData.push\_back(op);  }  else{  pData.push\_back(op);  pData.push\_back(' ');  pData.push\_back('(');  pData.append(IntToString(number));  pData.push\_back(')');  }  return pData;  }  /\*  void Tree::Make(int left, int right, int result, char op){  cout << "left:" << left << "/right:" << right << "/result:" << result << "/op:" << op << endl;  }  \*/  void Tree::Make(int left, int right, int result, char op){  TreeNode\* resultNode, \*leftNode, \*rightNode;  bool Check = false;  // 스택이 비어있을 때  if (newStack->IsEmpty())  {  if (op == '~')  {  leftNode = new TreeNode();  resultNode = new TreeNode();    leftNode->Data = IntToString(left);  leftNode->LeftNode = NULL;  leftNode->RightNode = NULL;  resultNode->Data = GetResult(result, op);  resultNode->LeftNode = leftNode;  resultNode->RightNode = NULL;  }  else{    leftNode = new TreeNode();  rightNode = new TreeNode();  resultNode = new TreeNode();  leftNode->LeftNode = NULL;  leftNode->Data = IntToString(left);  leftNode->RightNode = NULL;  rightNode->LeftNode = NULL;  rightNode->Data = IntToString(right);  rightNode->RightNode = NULL;  resultNode->LeftNode = leftNode;  resultNode->Data = GetResult(result, op);  resultNode->RightNode = rightNode;  }  newStack->push(resultNode);  }  else  {  if (newStack->GetCount() > 1){  TreeNode\* pNode = newStack->GetTop();  TreeNode\* pLeftNode, \*pRightNode;  // '^'일 때  if (op == '^'){  leftNode = new TreeNode();  rightNode = new TreeNode();  resultNode = new TreeNode();  leftNode->LeftNode = NULL;  leftNode->Data = IntToString(left);  leftNode->RightNode = NULL;  rightNode->LeftNode = NULL;  rightNode->Data = IntToString(right);  rightNode->RightNode = NULL;  resultNode->LeftNode = leftNode;  resultNode->Data = GetResult(result, op);  resultNode->RightNode = rightNode;  newStack->push(resultNode);  Check = true;  // Push 구현  } // End of if  // 1-1. '~'일 때  if (op == '~')  {  leftNode = new TreeNode();  resultNode = new TreeNode();  leftNode->Data = IntToString(left);  leftNode->LeftNode = NULL;  leftNode->RightNode = NULL;  resultNode->Data = GetResult(result, op);  resultNode->LeftNode = leftNode;  resultNode->RightNode = NULL;  newStack->push(resultNode);  Check = true;  // Push 구현  } // End of if  // 1-2. '~'일 때  if (pNode->Data == "~"){  leftNode = new TreeNode();  rightNode = new TreeNode();  resultNode = new TreeNode();  leftNode = newStack->pop();  rightNode->LeftNode = NULL;  rightNode->Data = IntToString(right);  rightNode->RightNode = NULL;  resultNode->LeftNode = leftNode;  resultNode->Data = GetResult(result, op);  resultNode->RightNode = rightNode;  newStack->push(resultNode);  Check = true;  // Push 구현  } // End of if  // LeftNode - RightNode  if (Check == false){  int LeftNumber;  int RightNumber;  leftNode = newStack->pop();  rightNode = newStack->pop();  LeftNumber = resultToInt(rightNode->Data);  RightNumber = resultToInt(leftNode->Data);  if (LeftNumber == left && RightNumber == right)  {  resultNode = new TreeNode();  resultNode->Data = GetResult(result, op);  resultNode->LeftNode = rightNode;  resultNode->RightNode = leftNode;  newStack->push(resultNode);  }  else  {  newStack->push(rightNode);  newStack->push(leftNode);  leftNode = new TreeNode();  rightNode = new TreeNode();  resultNode = new TreeNode();  leftNode->LeftNode = NULL;  leftNode->Data = IntToString(left);  leftNode->RightNode = NULL;  rightNode->LeftNode = NULL;  rightNode->Data = IntToString(right);  rightNode->RightNode = NULL;  resultNode->LeftNode = leftNode;  resultNode->Data = GetResult(result, op);  resultNode->RightNode = rightNode;  newStack->push(resultNode);  } // End of if  } // End of if  }  else  {  // 1-1. '~'일 때  if (op == '~')  {  leftNode = new TreeNode();  resultNode = new TreeNode();  leftNode->Data = IntToString(left);  leftNode->LeftNode = NULL;  leftNode->RightNode = NULL;  resultNode->Data = GetResult(result, op);  resultNode->LeftNode = leftNode;  resultNode->RightNode = NULL;  newStack->push(resultNode);  Check = true;  // Push 구현  } // End of if  else  {  leftNode = new TreeNode();  rightNode = new TreeNode();  resultNode = new TreeNode();  leftNode->LeftNode = NULL;  leftNode->Data = IntToString(left);  leftNode->RightNode = NULL;  rightNode->LeftNode = NULL;  rightNode->Data = IntToString(right);  rightNode->RightNode = NULL;  resultNode->LeftNode = leftNode;  resultNode->Data = GetResult(result, op);  resultNode->RightNode = rightNode;  newStack->push(resultNode);  }    }  } // End of if  }  int Tree::resultToInt(std::string D){  int i = 0, j = 0;  int length = D.length();  int result = 0;  char ch;  bool Check = false;  bool Negative = false;  while (i < length)  {  ch = stringToChar(D, i);  // ) 일 때 종료처리  if (Check == true && ch != ')')  {  if (ch == '-')  {  Negative = true;  }  else  {  // 숫자 범위 파악  if (ch >= '0' && ch <= '9')  {  result = (result \* 10) + ch - '0';  } // End of if  } // End of if  } // End of if  if (ch == '('){  Check = true;  } // End of if  // Negative 수 처리  if (ch == ')'){  Check = false;  // 음수로 변환(Result)  if (Negative == true)  {  result = result \* (-1);  } // End of if  } // End of if  i++;  } // End of while  return result;  }  std::string IntToString(int number)  {  ostringstream convStream;  convStream << number;  return convStream.str();  }  std::string CharToString(char op){  string D;  D.push\_back(op);  return D;  }  const char stringToChar(std::string D, int i){  string T\_Data = D.substr(i, 1);  return T\_Data.at(0);  }  pStack::pStack(){  Count = 0;  Top = new pNode();  Top->Data = NULL;  Top->prev = NULL;  Top->next = NULL;  }  pStack::~pStack(){  pNode\* curNode;  while (Top != NULL)  {  curNode = Top;  Top = Top->prev;  delete curNode;  } // End of while  }  void pStack::push(TreeNode\* \_pNode){  pNode\* newNode = new pNode();  newNode->Data = \_pNode;  newNode->prev = NULL;  newNode->next = NULL;  // 신규 추가  if (Top->Data != NULL)  {  newNode->prev = Top;  Top->next = newNode;  Top = Top->next;  }  else  {  Top = newNode;  } // End of if  Count++;  }  TreeNode\* pStack::pop(){  TreeNode\* pTreeNode;  pNode\* pNode;  // 데이터가 없을 때  if (Top->Data == NULL)  return 0;  if (Top->prev != NULL)  {  pTreeNode = Top->Data;  pNode = Top;  Top = Top->prev;  delete pNode;  }  else  {  pTreeNode = Top->Data;  Top->Data = NULL;  Top->next = NULL;  } // End of if  Count--;  return pTreeNode;  }  TreeNode\* pStack::GetTop(){  return Top->Data;  }  int pStack::IsEmpty(){  if (Top->Data != NULL)  return 0;  else  return 1;  }  int pStack::GetCount(){  return Count;  }  PrintStack::PrintStack(){  Top = new PrintNode();  Top->Data = "NULL";  Top->prev = NULL;  Top->next = NULL;  }  PrintStack::~PrintStack(){  PrintNode\* curNode;  while (Top != NULL)  {  curNode = Top;  Top = Top->prev;  delete curNode;  } // End of while  }  void PrintStack::push(std::string \_Data, int \_Height){  PrintNode\* newNode = new PrintNode();  newNode->Data = \_Data;  newNode->Height = \_Height;  newNode->prev = NULL;  newNode->next = NULL;  // 신규 추가  if (Top->Data != "NULL")  {  newNode->prev = Top;  Top->next = newNode;  Top = Top->next;  }  else  {  Top->Data = \_Data;  Top->Height = \_Height;  } // End of if  Count++;  }  std::string PrintStack::pop(){  std::string pData;  PrintNode\* pNode;  // 데이터가 없을 때  if (Top->Data == "NULL")  return 0;  if (Top->prev != NULL)  {  pData = Top->Data;  pNode = Top;  Top = Top->prev;  delete pNode;  }  else  {  pData = Top->Data;  Top->Data = "NULL";  Top->next = NULL;  } // End of if  Count--;  return pData;  }  int PrintStack::GetCount(){  return Count;  }  std::string PrintStack::at(int height){  std::string pData = "NULL";  PrintNode\* pNode = Top;  while (1)  {  // 루프 탈출  if (pNode == NULL)  break;  if (pNode->Height == height){  pData = pNode->Data;  break;  }  pNode = pNode->prev;  }  return pData;  }  #endif |

**Figure 3) tree\_detail.h**

|  |
| --- |
| /\*  OOOOOOOOOOOOO교  과제: 방학(Binary Tree) 과제5 - 2번(계산식)  학과: Business & CSE  학번: 20101215  이름: 정OO  \*/  #ifndef \_STACK\_H\_  #define \_STACK\_H\_  #include <iostream>  #include <string>  using namespace std;  const int MAX = 100;  template <class T>  class BinaryNode;  template <class T>  class BinaryTree;  template <class T>  class CalcTree;  template <class T>  class Node{  public:  T Data;  Node<T>\* prev;  Node<T>\* next;  };  template <class T>  class Stack{  public:  Stack();  ~Stack();  void push(T t);  T pop(void);  T get\_stack\_top(void);  int is\_stack\_empty(void);  private:  int Count;  Node<T>\* Top;  };  #include "stack\_inl.h"  #endif |

**Figure 4) stack.h**

|  |
| --- |
| /\*  금오공과대학교  과제: 방학(Binary Tree) 과제5 - 2번(계산식)  학과: Business & CSE  학번: 20101215  이름: 정원영  \*/  #ifndef \_STACK\_INL\_H\_  #define \_STACK\_INL\_H\_  #include <iostream>  #include "stack.h"  using namespace std;  template <class T>  Stack<T>::Stack(){  Count = 0;  Top = new Node<T>();  Top->Data = NULL;  Top->prev = NULL;  Top->next = NULL;  }  template <class T>  Stack<T>::~Stack(){  Node<T>\* prevNode;  while (Top != NULL)  {  prevNode = Top;  Top = Top->prev;  delete prevNode;  }  }  template <class T>  void Stack<T>::push(T Data){  if (Top->Data == NULL)  {  Top->Data = Data;  }  else  {  Node<T>\* createNode = new Node<T>();  createNode->Data = Data;  createNode->prev = Top;  createNode->next = NULL;  Top->next = createNode;  Top = Top->next;  }  Count++;  }  template <class T>  T Stack<T>::pop(){  T Data = NULL;  if (Top->prev == NULL)  {  Top->prev = NULL;  Top->next = NULL;  Data = Top->Data;  Top->Data = NULL;  }  else{  Node<T>\* prevNode = Top->prev;  Data = Top->Data;  delete Top;  Top = prevNode;  }  Count--;  return Data;  }  template <class T>  T Stack<T>::get\_stack\_top(void)  {  return (Top->Data == NULL) ? -1 : Top->Data;  }  template <class T>  int Stack<T>::is\_stack\_empty(void)  {  return (Top->Data == NULL);  }  #endif |

**Figure 5) stack\_inl.h**