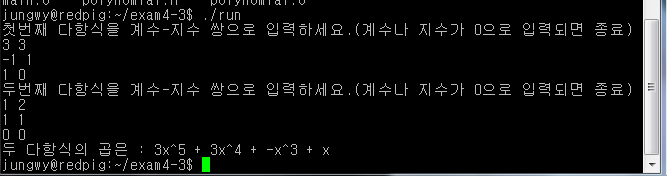


|  |
| --- |
| 다항식 곱셈 |
| **과제 4장 보고서(3) – HeadNode 제거** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **제 출 일** | **2014. 06. 07** |  | **전 공** | **Business**  **& Candidate CSE** |
| **과 목** | **자료구조론** |  | **학 번** | **20101215** |
| **담당교수** |  |  | **이 름** | **정 O O** |

**1. 다항식곱셈 출력 결과  
\* 동작 결과**

****

**2. 원리소개**

|  |
| --- |
| coef의 Node를 환형(Circular)구조로 구성하여 firstnode의 위치를 구했고, 더불어 Headnode를 제거하였다. |
|  |
| 계산원리는 신규 리스트에 먼저 담긴 것을 Exp가 일치한지 파악하고 그에 맞게 처리한다.  전체데이터를 다 찾지 않아도 되고 일치한 데이터 만큼 찾는데 바구니에 담긴 시간 대비로 보면 O() 시간이면 끝난다.  로 잡은 건 일치할 땐 바로 탈출할 수도 있는 상황들을 고려했다.  (순수하게 돌리면 걸리지만, 이에 비하면 엄청 시간이 줄어들었다고 볼 수 있다. |

**3. 코드 결과**

|  |
| --- |
| /\* OOOOOOOOOO교 경영학과  학번: 20101215  이름: 정 O O  과제: 4장 (3번) - 다항식 곱셈 (HeadNode 제거)  \*/  #include <iostream>  #include "polynomial.h"  using namespace std;  void PrintTitle(int);  int main()  {  PolyNode X, Y, Result;  List PolyList;  PolyNomial Poly;  cout << "첫번째";  PolyList.Write(X);  cout << "두번째";  PolyList.Write(Y);  Result = \*Poly.Multiply(&X, &Y);  cout << "두 다항식의 곱은 : ";  Poly.PrintOut(Result);  } |

Figure ) main.cpp

|  |
| --- |
| /\* OOOOOOOOOO교 경영학과  학번: 20101215  이름: 정 O O  과제: 4장 (3번) - 다항식 곱셈 (HeadNode 제거)  \*/  #ifndef \_POLYNOMIAL\_H\_  #define \_POLYNOMIAL\_H\_  const int EMPTY = -1;  enum VAR{ COEF, EXP };  class PolyNode{  public:  PolyNode();  ~PolyNode();  int Type;  int Coef;  PolyNode\* Prev;  PolyNode\* Next;  PolyNode\* Up;  PolyNode\* Down;    union  {  int Exp;  };    };  class List{  public:  List();  ~List();  void Write(PolyNode& List);  PolyNode\* Insert(PolyNode\* dSource, PolyNode\* dNode);  PolyNode\* Create(int \_Type, int \_Data);    void at(int \_IDX);  private:  };  class PolyNomial{  public:  PolyNomial();  ~PolyNomial();  PolyNode\* pAdd(PolyNode\* A\_List, PolyNode\* B\_List);  PolyNode\* Multiply(PolyNode\* A\_List, PolyNode\* B\_List);  void PrintOut(PolyNode List);  private:  };  #endif |

Figure ) polynomial.h

|  |
| --- |
| /\* OOOOOOOOOO교 경영학과  학번: 20101215  이름: 정 O O  과제: 4장 (3번) - 다항식 곱셈 (HeadNode 제거)  \*/  #include <iostream>  #include "polynomial.h"  using namespace std;  PolyNode::PolyNode(){  Type = -1;  }  PolyNode::~PolyNode(){  }  List::List(){  }  List::~List(){  }  void List::Write(PolyNode& List){    cout << " 다항식을 계수-지수 쌍으로 입력하세요.(계수나 지수가 0으로 입력되면 종료)" << endl;  PolyNode\* NewList = new PolyNode();  while (1)  {  int coefNum;  int expNum;    cin >> coefNum;  cin >> expNum;  if (coefNum != 0)  {  PolyNode\* CoefNode = Create(COEF, coefNum); // 계수 생성  PolyNode\* ExpNode = Create(EXP, expNum); // 지수 생성    NewList = Insert(CoefNode, NewList); // 계수 삽입  NewList = Insert(ExpNode, NewList); // 지수 삽입  }  if (coefNum == 0 || expNum == 0)  break;  cin.clear();  }  List = \*NewList;  }  PolyNode\* List::Create(int \_Type, int \_Data){  PolyNode\* NewList = new PolyNode();    NewList->Type = \_Type;  switch (\_Type)  {  case COEF:  NewList->Coef = \_Data;  NewList->Down = NULL;  break;  case EXP:  NewList->Up = NULL;  NewList->Exp = \_Data;  break;  }  NewList->Prev = NULL;  NewList->Next = NULL;  return NewList;  }  PolyNode\* List::Insert(PolyNode\* dSource, PolyNode\* dNode)  {  PolyNode\* TmpNode = dNode;  PolyNode\* PrevNode = TmpNode;  PolyNode\* FirstNode;  if (TmpNode->Type == EMPTY)  {  TmpNode = dSource;  TmpNode->Prev = TmpNode;  TmpNode->Next = TmpNode;  }  else  {  FirstNode = dNode->Next; // FirstNode 가리키기.(환형 리스트)  if (TmpNode->Up != NULL)  TmpNode = TmpNode->Up;  while (TmpNode != NULL)  {  PrevNode = TmpNode;  if (TmpNode->Next == FirstNode) // FirstNode에 도달했을 때 탈출.  break;  TmpNode = TmpNode->Next;  }  switch (dSource->Type)  {  case COEF:  dSource->Prev = PrevNode;  dSource->Next = FirstNode;  PrevNode->Next = dSource;  FirstNode->Prev = PrevNode->Next; // 첫위치 변경  TmpNode = dSource;  break;  case EXP:  dSource->Up = PrevNode;  PrevNode->Down = dSource;  TmpNode = PrevNode;  break;  }  }  return TmpNode;  }  PolyNomial::PolyNomial(){  }  PolyNomial::~PolyNomial(){  }  PolyNode\* PolyNomial::Multiply(PolyNode\* A\_List, PolyNode\* B\_List){  int CoefNum, ExpNum;  PolyNode\* X\_List,\* Y\_List;    PolyNode\* X\_NextNode, \*Y\_NextNode;    PolyNode\* TmpNode, \*FirstNode;  PolyNode\* NewNode = new PolyNode();  List PolyList;  X\_List = A\_List->Next; // A-List의 FirstNode 가리킴  Y\_List = B\_List->Next; // B-List의 FirstNode 가리킴  X\_NextNode = X\_List;    do  {  Y\_NextNode = Y\_List;  while (Y\_NextNode != NULL)  {  PolyNode\* NewCreate = new PolyNode();  // X - 계산  CoefNum = X\_NextNode->Coef;  TmpNode = X\_NextNode->Down;  ExpNum = TmpNode->Exp;  // Y - 계산  CoefNum \*= Y\_NextNode->Coef;  TmpNode = Y\_NextNode->Down;  ExpNum += TmpNode->Exp;  // X\*Y - COEF  TmpNode = PolyList.Create(COEF, CoefNum);  NewCreate = PolyList.Insert(TmpNode, NewCreate);    // X+Y - EXP  TmpNode = PolyList.Create(EXP, ExpNum);  NewCreate = PolyList.Insert(TmpNode, NewCreate);  NewNode = pAdd(NewCreate, NewNode);  // 다음 노드로 이동  Y\_NextNode = Y\_NextNode->Next;  if (Y\_NextNode == Y\_List) // Y\_list 첫 위치에 도달시 탈출  break;  }  X\_NextNode = X\_NextNode->Next;  // 다음 노드로 이동  if (X\_NextNode == X\_List) // X\_list 첫 위치에 도달시 탈출  break;    } while (X\_NextNode != NULL);  return NewNode;  }  PolyNode\* PolyNomial::pAdd(PolyNode \*A\_List, PolyNode \*B\_List){  PolyNode \*PrevNode, \*NextNode = B\_List;  PolyNode \*FirstNode, \*LeafNode;  int User\_Coef, User\_Exp;  List PolyList;    if (NextNode->Type == EMPTY)  {  NextNode = A\_List;  }  else  {  FirstNode = B\_List->Next;  NextNode = FirstNode;  PrevNode = NextNode;  User\_Coef = A\_List->Coef;  LeafNode = A\_List->Down;  User\_Exp = LeafNode->Exp;  while (NextNode != NULL)  {  LeafNode = NextNode->Down;  if (User\_Exp == LeafNode->Exp)  {  int Sum = NextNode->Coef + User\_Coef;  NextNode->Coef = Sum;  FirstNode = FirstNode->Prev;  NextNode = FirstNode->Prev;  break;  }  PrevNode = NextNode;  NextNode = NextNode->Next;    // NextNode가 FirstNode에 일치할 때  if (NextNode == FirstNode)  {  NextNode = PrevNode;  A\_List->Prev = NextNode;  A\_List->Next = FirstNode;  NextNode->Next = A\_List;  FirstNode->Prev = NextNode->Next;  break;  }    }  }  return NextNode->Next;  }  void PolyNomial::PrintOut(PolyNode List){  PolyNode\* FirstNode = List.Next;  PolyNode\* NextNode = List.Next;  PolyNode\* TmpNode;  int CoefNum, ExpNum;  while (NextNode != NULL)  {  if (NextNode->Coef != 0)  {  TmpNode = NextNode->Down;  CoefNum = NextNode->Coef;  ExpNum = TmpNode->Exp;  switch (CoefNum)  {  case -1:  cout << "-x";  break;    case 1:  cout << "x";  break;  default:  cout << CoefNum << "x";  break;  }  if (ExpNum != 1) // 지수가 1이 아닐 때  cout << "^" << ExpNum;    }  else{  NextNode = NextNode->Next;  continue;  }  if (NextNode->Next == FirstNode) // 목표지점 도달시 탈출  break;  else  cout << " + ";  NextNode = NextNode->Next;  }    cout << endl;  } |

Figure ) Polynomial.cpp