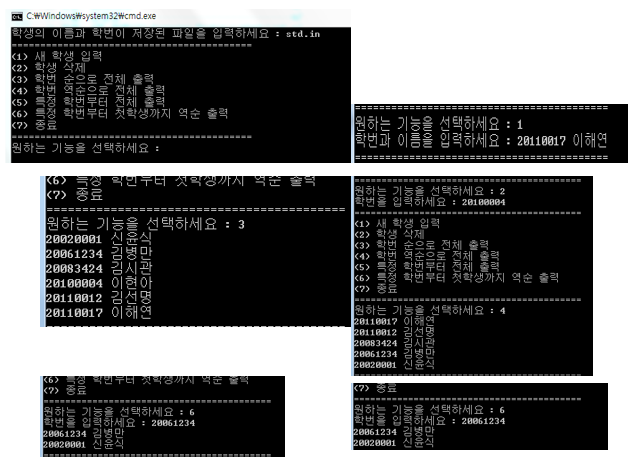


|  |
| --- |
| 이중연결원형리스트 |
| **과제 4장 보고서 – HeadNode 제거** |

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

**1. 이중연결원형리스트(출력결과)**

**동작 결과**



**2. 원리소개**

|  |
| --- |
| enum { tCode, tData }; // Field  class Node{  public:  int type;  int code;  std::string Data;  Node\* prev;  Node\* next;  Node\* up;  Node\* down;  }; |
| **Node의 구성**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Type** | **Code** | **Data** | **Prev** | **Next** | **Up** | **Down** | | **tCode** | **학번** | **“”** | **최근 데이터(자기자신도 포함)** | **다음 노드** | **학번을 기준으로 하는 세부 데이터 중 최근 데이터**  **(또는 이전데이터)** | **이전 데이터 (자기자신도 포함)** |     **이와 같은 원리로 구성하면 Top 필요 없이 마지막 Down과 Next를 통해 First로 갈 수 있음.**  **환형원형연결리스트의 원리 적용** |

**3. 코드 결과**

|  |
| --- |
| /\* OOOOOOOOOO \*/  /\* 학번: 20101215 \*/  /\* 이름: 정OO \*/  /\* 과제: 4장 과제(2) Circular Double Linked List - HeadNode 존재 \*/  #include <iostream>  #include "ddl.h"  const int NEW = 1;  const int REMOVE = 2;  const int NORMAL\_ASC = 3;  const int NORMAL\_DESC = 4;  const int SPECIFIC\_ASC = 5;  const int SPECIFIC\_DESC = 6;  const int EXIT = 7;  using namespace std;  int main()  {  TManager tm;  Node dNode;  bool inFile = tm.LoadFile(&dNode); // 파일 입력 받기  while (inFile)  {  int menu;  tm.PrintMenu(); // 타이틀 출력  cin >> menu; // 기능 입력 받기    if (menu == EXIT)  break;  switch (menu)  {  case NEW:  tm.New(&dNode);  break;  case REMOVE:  tm.Remove(&dNode);  break;  case NORMAL\_ASC:  tm.NormalAsc(dNode);  break;  case NORMAL\_DESC:  tm.NormalDesc(dNode);  break;  case SPECIFIC\_ASC:  tm.SpecificAsc(dNode);  break;  case SPECIFIC\_DESC:  tm.SpecificDesc(dNode);  break;    default:  cout << "기능을 올바르게 입력하세요.";  break;  }  }  return 0;  } |

Figure ) main.cpp

|  |
| --- |
| /\* OOOOO \*/  /\* 학번: 20101215 \*/  /\* 이름: 정OO \*/  /\* 과제: 4장 과제(2) Circular Double Linked List \*/  #ifndef \_DDL\_H\_  #define \_DDL\_H\_  #include <iostream>  #include <string>  // 출력  const int REVERSE = -1;  const int NONE = 0;  // 정렬 방법  const int ASC = 1;  const int DESC = 2;  // 조회 타입  const int NORMAL = 3; // 보통 조회  const int SPECIFIC = 4; // 특정 학번 조회  // Column MAX  const int MAXCOLUMNS = 2; // COLUMNS MAX  enum { tCode, tData }; // Field  class Node{  public:  int type;  int code;  std::string Data;  Node\* prev;  Node\* next;  Node\* up;  Node\* down;  };  class DDList{  public:  DDList();  ~DDList();  Node\* Create(int \_Type, std::string Data);  Node\* Insert(Node\* createNode, Node\* pNode);  Node\* Delete(int \_Code, Node\* pNode);  };  class TManager{  friend void PrintTitle();  public:  TManager();  ~TManager();  void PrintMenu();  bool LoadFile(Node\* pNode);  void New(Node\* pNode);  void Remove(Node\* pNode);  void NormalAsc(Node pNode);  void NormalDesc(Node pNode);  void SpecificAsc(Node pNode);  void SpecificDesc(Node pNode);  };  class IOHandler{  public:  IOHandler();  ~IOHandler();  bool InputFile(std::string Data, Node& pNode);  /\*  Default(특정 지정 없이 출력 가능)  설명: (DDList\* List,  int Type = NORMAL, ,  int SortType = ASC, DESC  int Type = SPECIFIC  int SortType = NONE, NORMAL, REVERSE,  int Code = 코드(학번) 입력  \*/  void PrintOut(Node pNode, int PrintType = NORMAL, int SortType = ASC, int Code = NONE);  private:  void UserView(Node\* pNode);  void Print\_Normal(Node\* pNode, int SortType);  void Print\_Specific(Node\* pNode, int Sort, int Code);  };  class Algorithm{  public:  Node\* Sort(Node\* pNode);  private:  void Swap(Node\* X\_Node, Node\* Y\_Node);  };  #endif |

Figure ) ddl.h

|  |
| --- |
| /\* OOOOOOOOOOOOO \*/  /\* 학번: 20101215 \*/  /\* 이름: 정OO \*/  /\* 과제: 4장 과제(2) Circular Double Linked List \*/  #include <iostream>  #include <string>  #include "ddl.h"  using namespace std;  Node\* Algorithm::Sort(Node\* pNode){  Node newNode = \*pNode;  Node\* FirstNode = newNode.down;  FirstNode = FirstNode->next;  Node\* X\_Node = newNode.down;  Node\* Y\_Node = NULL, \*MinNode = NULL, TmpNode;  // 버블 정렬  while (X\_Node != NULL)  {  Y\_Node = FirstNode;  while (Y\_Node != NULL)  {  if (X\_Node->code > Y\_Node->code){  Swap(X\_Node, Y\_Node);  }  if (Y\_Node->next == FirstNode)  break;  Y\_Node = Y\_Node->next;    } // end of while  if (X\_Node->next == FirstNode)  break;  X\_Node = X\_Node->next;    }// end of while  return X\_Node;  }  void Algorithm::Swap(Node\* X\_Node, Node\* Y\_Node){  int TmpNum;  string TmpTxt;  Node\* FirstNode = X\_Node;  TmpNum = X\_Node->code;  X\_Node->code = Y\_Node->code;  Y\_Node->code = TmpNum;  Node\* X\_DownNode = X\_Node->down;  Node\* Y\_DownNode = Y\_Node->down;  while (X\_DownNode != NULL) // 하위 항목까지  {  TmpTxt = X\_DownNode->Data;  X\_DownNode->Data = Y\_DownNode->Data;  Y\_DownNode->Data = TmpTxt;  if (X\_DownNode->down == FirstNode)  break;    X\_DownNode = X\_DownNode->down; // 참고X\_DownNode나 Y\_DownNode나 아래 깊이는 같음.  Y\_DownNode = Y\_DownNode->down;  }  } |

Figure ) algorithm.cpp

|  |
| --- |
| /\* OOOOOOOOOOOO \*/  /\* 학번: 20101215 \*/  /\* 이름: 정OO \*/  /\* 과제: 4장 과제(2) Circular Double Linked List \*/  #include <iostream>  #include <string>  #include <direct.h>  #include "ddl.h"  using namespace std;  DDList::DDList(){  }  DDList::~DDList(){  }  Node\* DDList::Insert(Node\* createNode, Node\* pNode){  Node\* PrevNode, \*FirstNode;  Node\* NextNode = pNode;  Node\* CurNode;  if (pNode->up == NULL)  {  pNode = createNode;  pNode->prev = pNode;  pNode->next = pNode;  pNode->up = pNode;  pNode->down = pNode;  }  else  {  if (createNode->type == tCode)  {  FirstNode = pNode->down;  FirstNode = FirstNode->next;  PrevNode = pNode->down;  PrevNode->next = createNode;    CurNode = PrevNode->next;  FirstNode->prev = CurNode;  CurNode->prev = PrevNode;  CurNode->next = FirstNode;  CurNode->up = CurNode;  CurNode->down = CurNode;  pNode = CurNode;  }  else  {  FirstNode = NextNode->down;  PrevNode = pNode->up;  createNode->down = FirstNode;  createNode->up = PrevNode;  FirstNode->up = createNode;  PrevNode->down = FirstNode->up;  pNode = PrevNode->down;  }  }  return pNode;  }  Node\* DDList::Delete(int \_Code, Node\* pNode){  Node\* CurNode;  Node\* FirstNode = pNode->down;  FirstNode = FirstNode->next;  Node\* NextNode = pNode->down;  NextNode = NextNode->next;  // 환형 더블리스트이므로 마지막 부위가 First인점  Node\* PrevNode = NextNode;  while (NextNode != NULL)  {  if (\_Code == NextNode->code)  {  PrevNode->next = NextNode->next;  CurNode = NextNode;  NextNode = NextNode->next;  NextNode->prev = PrevNode;    delete CurNode;  break;  }  PrevNode = NextNode;  NextNode = NextNode->next;  // NextNode가 First에 도달할 때  if (NextNode->code == FirstNode->code)  break;    }  return NextNode;  }  Node\* DDList::Create(int \_Type, string Data){    Node\* CreateNode = new Node();  int Num;  switch (\_Type)  {  case tCode:  Num = atoi(Data.c\_str());  CreateNode->type = tCode;  CreateNode->code = Num;  CreateNode->up = NULL;  CreateNode->down = NULL;  CreateNode->prev = NULL;  CreateNode->next = NULL;  break;  case tData:  CreateNode->type = tData;  CreateNode->Data = Data;  CreateNode->up = NULL;  CreateNode->down = NULL;  CreateNode->prev = NULL;  CreateNode->next = NULL;  break;  }  return CreateNode;  } |

Figure 4) ddl.cpp

|  |
| --- |
| /\* OOOOOOOOOOOOOO \*/  /\* 학번: 20101215 \*/  /\* 이름: 정OO \*/  /\* 과제: 4장 과제(2) Circular Double Linked List \*/  #include <iostream>  #include <direct.h>  #include "ddl.h"  using namespace std;  IOHandler::IOHandler(){  }  IOHandler::~IOHandler(){  }  bool IOHandler::InputFile(string Data, Node& pNode){  bool Check = true;  int ColumnIDX = 1;  int i = 0;  int MaxSize = Data.size();  string chData, TmpData;  Node\* CreateNode;  DDList dList;  while (i < MaxSize)  {  chData = Data.substr(i, 1);  if (chData == " ") // 다음 컬럼 넘기기  {  if (ColumnIDX == 1) // Column은 고유 코드 입력(숫자형)  {  CreateNode = dList.Create(tCode, TmpData);  }  else  {  CreateNode = dList.Create(tData, TmpData);  }  pNode = \*dList.Insert(CreateNode, &pNode);  TmpData.erase(); // 임시 공간 초기화  ColumnIDX++;  }  if (ColumnIDX == 1) // 첫 코드에 대한 데이터 임시 Append(추가)  {  if (isdigit(chData.at(0)))  TmpData.append(chData.c\_str());  else{ Check = false; break; }  }  else  {  if (chData != " " && chData != "\n") // 공백이 아닐 때  TmpData.append(chData.c\_str());  }  // 단 넘어갈 때 처리  if (chData.at(0) == '\n')  {  if (ColumnIDX > MAXCOLUMNS) // 최대 범위를 넘어갈 때  {  Check = false;  break;  }  if (ColumnIDX == MAXCOLUMNS) // 데이터가 일치할 때  {  CreateNode = dList.Create(tData, TmpData);  pNode = \*dList.Insert(CreateNode, &pNode);  TmpData.erase(); // 임시 공간 초기화  }  if (ColumnIDX < MAXCOLUMNS) // 최소 범위를 넘어갈 때  {  Check = false;  break;  }  ColumnIDX = 1; // 컬럼 초기화  }  i++;  }  return Check;  }  void IOHandler::PrintOut(Node pNode, int PrintType, int SortType, int Code){  switch (PrintType)  {  case NORMAL:  Print\_Normal(&pNode, SortType);  break;  case SPECIFIC:  Print\_Specific(&pNode, SortType, Code);  break;  }  }  void IOHandler::Print\_Normal(Node\* pNode, int SortType){  Node\* FirstNode;  Node\* PrevNode;  Node\* NextNode;    switch (SortType)  {  case ASC:  FirstNode = pNode;  PrevNode = FirstNode;  while (PrevNode != NULL)  {  UserView(PrevNode); // 출력  PrevNode = PrevNode->prev;  if (PrevNode->code == FirstNode->code)  break;  }  break;  case DESC:  FirstNode = pNode->next;  NextNode = FirstNode;  while (NextNode != NULL)  {  UserView(NextNode); // 출력  NextNode = NextNode->next;  if (NextNode->code == FirstNode->code)  break;  }  break;  }  }  void IOHandler::Print\_Specific(Node\* pNode, int Sort, int Code){    Node\* FirstNode = pNode;  Node\* NextNode = FirstNode;    bool Escape = false;  bool Circular = false;  while (NextNode != NULL)  {  switch (Sort)  {  case NORMAL:    if (NextNode->code == Code)  {  if (Circular == false)  Circular = true;  else  {  Circular = false;  Escape = true;  }  }  // 데이터 출력  if (Circular == true)  UserView(NextNode);  NextNode = NextNode->prev;  break;  case REVERSE:  if (NextNode->code == Code)  {  if (Circular == false)  Circular = true;  }  // HeadNode(First)에 도달했을 때, Circular가 Prev로 전환된 상태  if (NextNode->code == FirstNode->code && Circular == true)  Escape = true;  if (Circular == false)  NextNode = NextNode->prev;  else{  UserView(NextNode); // 출력  NextNode = NextNode->next;  }  break;  }  if (Escape == true) // 탈출  break;  }  }  void IOHandler::UserView(Node\* pNode){    Node\* FirstNode = pNode;  Node\* DownNode;  cout << pNode->code << " ";  DownNode = pNode->down;  while (DownNode != NULL)  {  cout << DownNode->Data << endl;  DownNode = DownNode->down;  if (DownNode->code == FirstNode->code)  break;  }  } |

Figure 5) iohandler.cpp

|  |
| --- |
| /\* OOOOOOOOOOOOOO \*/  /\* 학번: 20101215 \*/  /\* 이름: 정OO \*/  /\* 과제: 4장 과제(2) Circular Double Linked List \*/  #include <iostream>  #include <fstream>  #include <direct.h>  #include "ddl.h"  using namespace std;  TManager::TManager(){  }  TManager::~TManager(){  }  void PrintTitle(){  cout << "학생의 이름과 학번이 저장된 파일을 입력하세요 : ";  }  void TManager::PrintMenu(){  cout << "==========================================" << endl;  cout << "(1) 새 학생 입력" << endl;  cout << "(2) 학생 삭제" << endl;  cout << "(3) 학번 순으로 전체 출력" << endl;  cout << "(4) 학번 역순으로 전체 출력" << endl;  cout << "(5) 특정 학번부터 전체 출력" << endl;  cout << "(6) 특정 학번부터 첫학생까지 역순 출력" << endl;  cout << "(7) 종료" << endl;  cout << "==========================================" << endl;  cout << "원하는 기능을 선택하세요 : ";  }  bool TManager::LoadFile(Node\* pNode){  bool Check = false;  ifstream Read;  IOHandler io;  string UserTxt, FileName, Tmp, Memo;  char\* Buff = \_getcwd(NULL, 0); // 현재경로 가져오기  PrintTitle(); // 타이틀 출력 - friend 함수  cin.sync();  std::getline(cin, UserTxt);  FileName.append(Buff); // Windows 기반 테스트  FileName.append("\\");  FileName.append(UserTxt);  Buff = &FileName[0];  Read.open(Buff);  while (getline(Read, Tmp))  {  Memo.append(Tmp);  Memo.append("\n");  Check = true;  }  if (Check == true)  Check = io.InputFile(Memo, \*pNode);  return Check;  }  void TManager::New(Node\* pNode){    DDList dList;  Node\* FirstNode = pNode;  Node\* CreateNode;  string UserTxt;  string chData, TmpData;  bool Check = false;  int i, Size, ColumnIDX;  while (1)  {  cout << "학번과 이름을 입력하세요 : ";  cin.sync();  std::getline(cin, UserTxt);    Size = UserTxt.size();    ColumnIDX = 1;  i = 0;  while (i < Size)  {  chData = UserTxt.substr(i, 1);  if (chData == " ")  {  if (ColumnIDX == 1)  CreateNode = dList.Create(tCode, TmpData);  else  CreateNode = dList.Create(tData, TmpData);  pNode = dList.Insert(CreateNode, pNode);  TmpData.erase();  ColumnIDX++;  }  if (ColumnIDX == 1)  {  if (isdigit(chData.at(0)))  TmpData.append(chData.c\_str());  else  break;  }  else  {  if (chData != " ")  TmpData.append(chData.c\_str());  }  if ((i + 1) == Size)  {  if (ColumnIDX > MAXCOLUMNS) // 최대 범위를 넘어갈 때  break;  if (ColumnIDX == MAXCOLUMNS) // 데이터가 일치할 때  {  CreateNode = dList.Create(tData, TmpData);  pNode = dList.Insert(CreateNode, pNode);  Check = true;  break;  }  if (ColumnIDX < MAXCOLUMNS) // 최소 범위를 넘어갈 때  break;  }  i++;  }  // 조건이 맞을 때  if (Check == true){  //pNode = dList.Insert(FirstNode, pNode); // 삽입  break;  }  else{  TmpData.erase(); // 임시 공간 초기화  }  UserTxt.erase(); // 입력 데이터 초기화  }  }  void TManager::Remove(Node\* pNode){  DDList dList;  int Code;  cout << "학번을 입력하세요 : ";  cin >> Code;  pNode = dList.Delete(Code, pNode);  }  void TManager::NormalAsc(Node pNode){  Algorithm Algo;  IOHandler io;  Node\* dNode = Algo.Sort(&pNode);  io.PrintOut(\*dNode, NORMAL, ASC);  }  void TManager::NormalDesc(Node pNode){  Algorithm Algo;  IOHandler io;  Node\* dNode = Algo.Sort(&pNode);  io.PrintOut(\*dNode, NORMAL, DESC);  }  void TManager::SpecificAsc(Node pNode){  Algorithm Algo;  IOHandler io;  int Code;  cout << "학번을 입력하세요 : ";  cin >> Code;    Node\* dNode = Algo.Sort(&pNode);  io.PrintOut(\*dNode, SPECIFIC, NORMAL, Code);  }  void TManager::SpecificDesc(Node pNode){  Algorithm Algo;  IOHandler io;  int Code;  cout << "학번을 입력하세요 : ";  cin >> Code;  Node\* dNode = Algo.Sort(&pNode);  io.PrintOut(\*dNode, SPECIFIC, REVERSE, Code);  } |

Figure 6) tmanager.cpp