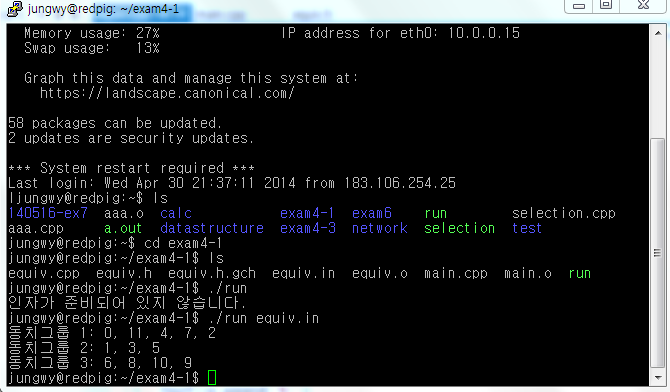


|  |
| --- |
| 동치관계 |
| **과제 4장 보고서(1)** |

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

**1. 동치관계**



**동작 결과**

**2. 분석 자료**

|  |
| --- |
| **12**  **0 4**  **3 1**  **6 10**  **8 9**  **7 4**  **6 8**  **3 5**  **2 11**  **11 0** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | int i, j, n;  inFile >> n;  ListNodePtr \*seq = new ListNodePtr[n];  Boolean \*out = new Boolean[n];  for (i = 0; i < n; i++) {  seq[i] = 0;  out[i] = FALSE;  } |  |  | | --- | | Infile을 통해 12를 읽는다.  Seq[0~12]까지 0으로 초기화한다.  Out[0~12]까지 FALSE로 초기화한다. |  |  | | --- | | // 1. 동치관계 데이터 읽어 들이기  inFile >> i >> j;  while (inFile.good()) // 파일 끝까지 찾기  {  ListNode \*x = new ListNode(j);  x->link = seq[i];  seq[i] = x; // 추가 (j부터 seq[i])    ListNode \*y = new ListNode(i);  y->link = seq[j];  seq[j] = y; // 추가 (i부터 seq[j])  inFile >> i >> j;  } |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Seq[0~12]도 노드의 공간(구조체로 표현 정리된)이다.  **먼저 x 노드(j)에 대해 설명하고자 한다. = j =4**   |  |  |  | | --- | --- | --- | | x | data | Link | | 4 | NULL |   **x->link 에 seq[i]를 대입**   |  |  |  | | --- | --- | --- | | x | Data | Link | | 4 | seq[0] |   **seq[0] = x주소를 대입**   |  |  |  | | --- | --- | --- | | Seq[0] | Data | Link | | 4 | Seq[0] |   **다음은 y 노드(i)에 대해 설명하고자 한다. i = 0**   |  |  |  | | --- | --- | --- | | Y | data | Link | | 0 | NULL |   **y->link 에 seq[j]를 대입**   |  |  |  | | --- | --- | --- | | x | Data | Link | | 0 | seq[4] |   **seq[4] = x주소를 대입**   |  |  |  | | --- | --- | --- | | Seq[4] | Data | Link | | 0 | Seq[4] |   **이와 같은 원리로 11, 0까지 저장된다.**   |  | | --- | | for (i = 0; i < n; i++)  {  if (out[i] == FALSE) { // 출력에 관한 부분  cout << "동치그룹 " << Cnt++ << ": " << i;  out[i] = TRUE;  ListNode \*x = seq[i];  ListNode \*top = 0; // 초기화 Stack  while (1) // 존재하지 않은 클래스 찾기  {  while (x) { // 프로세스 리스트 찾아내기  j = x->data;    if (out[j] == FALSE) {  cout << ", " << j;  out[j] = TRUE;  ListNode \*y = x->link;  x->link = top;  top = x;  x = y;  }  else  x = x->link;  } // While문 끝  if (!top)  break;  else {  x = seq[top->data];  top = top->link; // UnStack의 원리  }  }  cout << endl;  } // if문의 끝  }  } |   **Seq의 노드가 없을 때를 찾아 seq[top->data의 데이터]을 가리키면서 계속 돌면서**  **Top을 최후에는 Null로 도달하도록 하는 원리로서 출력한다.**  **자세한 로직 설명은 생략한다.** | |

**3. 코드 결과**

|  |
| --- |
| #include <iostream>  #include "equiv.h"  using namespace std;  void Equivalence(char\*);  // 인자 존재 여부(파일 이름 파악)  bool IsArgv(int argc, char\*\* argv){  int i = 0;  while (i < argc)  {  if (i > 1)  break;  i++;  }  if (i == 2)  return true;  else  return false;  }  char\* GetFileName(char\*\* argv){  return argv[1];  }  int main(int argc, char\*\* argv)  {  if (IsArgv(argc, argv) == true)  Equivalence(GetFileName(argv));  else  cout << "인자가 준비되어 있지 않습니다." << endl;    } |

Figure 1) Main.cpp

|  |
| --- |
| #ifndef \_EQUIV\_H\_  #define \_EQUIV\_H\_  #include <iostream>  #include <fstream>  enum Boolean { FALSE, TRUE };  class ListNode {  friend void Equivalence(char\*);  friend void PrintOut(ListNode\*\* seq, Boolean\* out, int n);  private:  int data;  ListNode \*link;  ListNode(int);  };  typedef ListNode\* ListNodePtr;  #endif |

Figure 2) Equiv.h

|  |
| --- |
| #include <iostream>  #include <fstream>  #include <unistd.h>  #include "equiv.h"  using namespace std;  ListNode::ListNode(int d)  {  data = d;  link = 0;  }  void Equivalence(char\* Filename)  {  ifstream inFile;  string YourFile;  char\* Buff = getcwd(NULL, 0); // 현재경로 가져오기  YourFile.append(Buff);  YourFile.append("\/");  YourFile.append(Filename);  Buff = &YourFile[0];  inFile.open(Buff);  if (!inFile)  {  cerr << "파일을 열수 없습니다. " << endl;  return;  }  int i, j, n;  inFile >> n;  ListNodePtr \*seq = new ListNodePtr[n];  Boolean \*out = new Boolean[n];  for (i = 0; i < n; i++) {  seq[i] = 0;  out[i] = FALSE;  }  // 1. 동치관계 데이터 읽어 들이기  inFile >> i >> j;  while (inFile.good()) // 파일 끝까지 찾기  {  ListNode \*x = new ListNode(j);  x->link = seq[i];  seq[i] = x; // 추가 (j부터 seq[i])    ListNode \*y = new ListNode(i);  y->link = seq[j];  seq[j] = y; // 추가 (i부터 seq[j])  inFile >> i >> j;  }  PrintOut(seq, out, n);  delete[] seq;  delete[] out;  }  // 2. 출력 클래스  void PrintOut(ListNode\*\* seq, Boolean\* out, int n){  int i, j;  int Cnt = 1;  for (i = 0; i < n; i++)  {  if (out[i] == FALSE) { // 출력에 관한 부분  cout << "동치그룹 " << Cnt++ << ": " << i;  out[i] = TRUE;  ListNode \*x = seq[i];  ListNode \*top = 0; // 초기화 Stack  while (1) // 존재하지 않은 클래스 찾기  {  while (x) { // 프로세스 리스트 찾아내기  j = x->data;    if (out[j] == FALSE) {  cout << ", " << j;  out[j] = TRUE;  ListNode \*y = x->link;  x->link = top;  top = x;  x = y;  }  else  x = x->link;  } // While문 끝  if (!top)  break;  else {  x = seq[top->data];  top = top->link; // UnStack의 원리  }  }  cout << endl;  } // if문의 끝  }  } |

Figure 3) Equiv.cpp