**홉필드 네트워크 리포트**

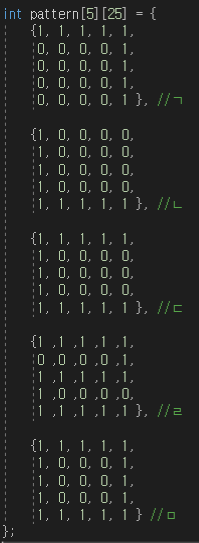
**이름 : 박건호**

**학과 : 컴퓨터공학과**

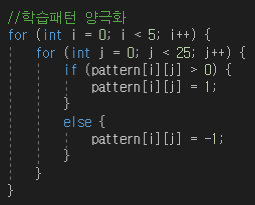
**학번 : 201635964**

**[알고리즘 순서]**

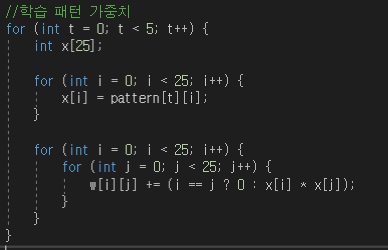
1. 학습할 패턴



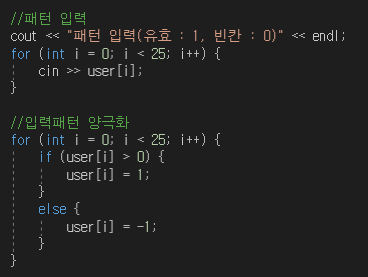
2. 학습패턴 양극화



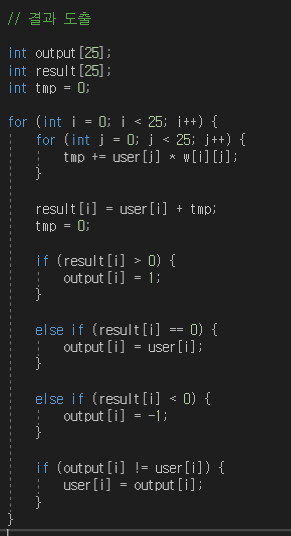
3. 학습패턴 가중치 계산



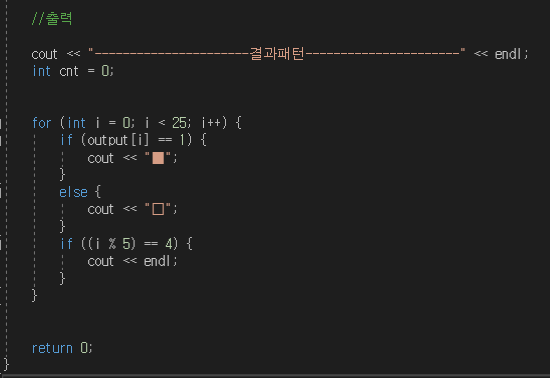
4. 패턴 입력 및 양극화



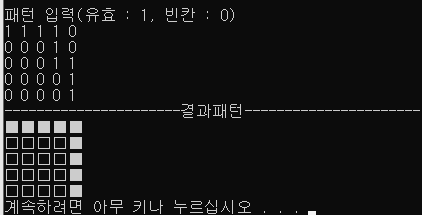
5. 결과 값 계산



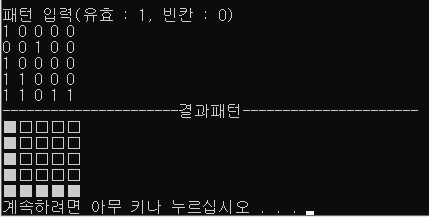
6. 결과 값 출력



**[예시 - ㄱ]**



**[예시 - ㄴ]**



**[전체코드]**

|  |
| --- |
| #include <iostream>  using namespace std;  int pattern[5][25] = {  {1, 1, 1, 1, 1,  0, 0, 0, 0, 1,  0, 0, 0, 0, 1,  0, 0, 0, 0, 1,  0, 0, 0, 0, 1 }, //ㄱ  {1, 0, 0, 0, 0,  1, 0, 0, 0, 0,  1, 0, 0, 0, 0,  1, 0, 0, 0, 0,  1, 1, 1, 1, 1 }, //ㄴ  {1, 1, 1, 1, 1,  1, 0, 0, 0, 0,  1, 0, 0, 0, 0,  1, 0, 0, 0, 0,  1, 1, 1, 1, 1 }, //ㄷ  {1 ,1 ,1 ,1 ,1,  0 ,0 ,0 ,0 ,1,  1 ,1 ,1 ,1 ,1,  1 ,0 ,0 ,0 ,0,  1 ,1 ,1 ,1 ,1 }, //ㄹ  {1, 1, 1, 1, 1,  1, 0, 0, 0, 1,  1, 0, 0, 0, 1,  1, 0, 0, 0, 1,  1, 1, 1, 1, 1 } //ㅁ  };  int user[25]; //입력값  int w[25][25]; //가중치  int main() {  cout << "----------------------학습패턴----------------------" << endl;  for (int i = 0; i < 5; i++) {  for (int j = 0; j < 25; j++) {  cout << pattern[i][j] << ' ';  if ((j % 5) == 4) {  cout << endl;  }  }  cout << endl;  }  //학습패턴 양극화  for (int i = 0; i < 5; i++) {  for (int j = 0; j < 25; j++) {  if (pattern[i][j] > 0) {  pattern[i][j] = 1;  }  else {  pattern[i][j] = -1;  }  }  }  cout << "----------------------학습패턴(양극화후)----------------------" << endl;  for (int i = 0; i < 5; i++) {  for (int j = 0; j < 25; j++) {  cout << pattern[i][j] << ' ';  if ((j % 5) == 4) {  cout << endl;  }  }  cout << endl;  }  //학습 패턴 가중치  for (int t = 0; t < 5; t++) {  int x[25];  for (int i = 0; i < 25; i++) {  x[i] = pattern[t][i];  }  for (int i = 0; i < 25; i++) {  for (int j = 0; j < 25; j++) {  w[i][j] += (i == j ? 0 : x[i] \* x[j]);  }  }  }  //패턴 입력  cout << "패턴 입력(유효 : 1, 빈칸 : 0)" << endl;  for (int i = 0; i < 25; i++) {  cin >> user[i];  }  //입력패턴 양극화  for (int i = 0; i < 25; i++) {  if (user[i] > 0) {  user[i] = 1;  }  else {  user[i] = -1;  }  }  // 결과 도출  int output[25];  int result[25];  int tmp = 0;  for (int i = 0; i < 25; i++) {  for (int j = 0; j < 25; j++) {  tmp += user[j] \* w[i][j];  }  result[i] = user[i] + tmp;  tmp = 0;  if (result[i] > 0) {  output[i] = 1;  }  else if (result[i] == 0) {  output[i] = user[i];  }  else if (result[i] < 0) {  output[i] = -1;  }  if (output[i] != user[i]) {  user[i] = output[i];  }  }  //출력  cout << "----------------------결과패턴----------------------" << endl;  int cnt = 0;    for (int i = 0; i < 25; i++) {  if (output[i] == 1) {  cout << "■";  }  else {  cout << "□";  }  if ((i % 5) == 4) {  cout << endl;  }  }  return 0;  } |

**[소감]**

홉필드 네크워크를 구현해보기 전에는 과연 이 알고리즘이 내가 구현해 낼 수 있는 알고리즘인가 라는 생각이 우선 들었다. 간단한 알고리즘을 통하여 문제 해결 방식의 코드만 작성해왔기 때문이다. 수업만으로 이 알고리즘을 완전히 이해하기 힘들었기 때문에 구현하기 전에 홉필드 알고리즘에 대하여 추가적으로 공부를 하였다. 공부를 하다보니 내가 생각한 것 만큼 복잡한 구조가 아니였고 행렬 계산식만 구현을 잘 한다면 문제 없을 것이라 생각하여 바로 구현에 들어갔다. 패턴을 학습시키고 그 패턴에 대한 가중치를 계산하여 입력된 패턴에 대한 학습패턴을 연상시켰다. 구현을 마친 후에는 만족스럽기 보다는 노가다(?) 형식으로 되어있는 코드가 마음에 들지 않아 조금씩 수정해가며 가독성을 높이고 복잡도를 낮추려고 노력했다. 물론 만족스러울 정도로 구현이 깔끔하고 완벽하게 되지는 않았지만 새로운 것을 하나씩 배운다는 것이 재미있었고 앞으로 문제를 해결해 나갈 때 어렵다고 포기하기 보다는 부딪혀보는게 좋을 것 같다. 홉필드 네트워크처럼 막상 구현하고 보면 그다지 복잡하고 어렵지 않은 알고리즘인 것 처럼 말이다.