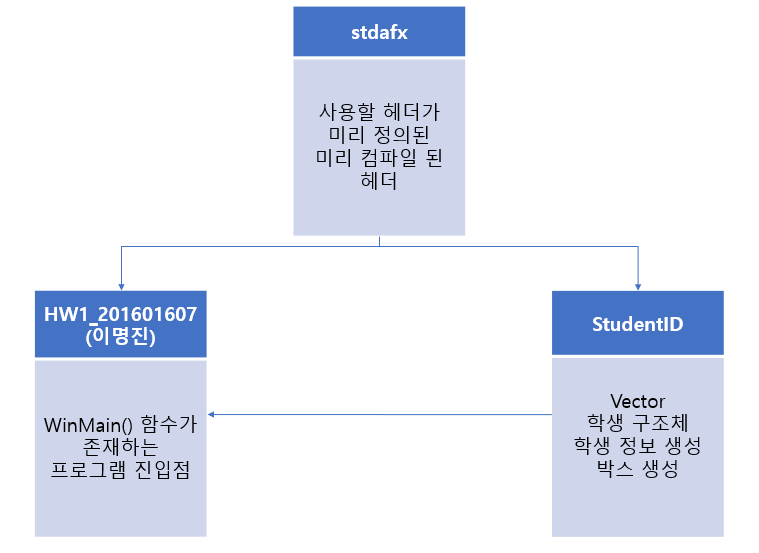
|  |
| --- |
| **게임 프로그래밍** |

- 과제 1 -

|  |  |
| --- | --- |
| 과목 | 게임 프로그래밍 |
| 교수 | 박종승 교수님 |
| 제출일 | 2021.03.26 |
| 학번 | 201601607 |
| 학부 | 컴퓨터공학부 |
| 이름 | 이명진 |
| E-mail | mpsa001@naver.com |
| 연락처 | 010-9929-6564 |

# 1: 코드 구조



stdafx(미리 컴파일된 헤더)

|  |
| --- |
| 미리 컴파일된 헤더 파일로 프로그램 작성에 필요한 헤더가 포함 |
| #include <iostream>  #include <tchar.h> //유니코드를 사용하기 위한 헤더  #include <string>  #include <ctime>;  //vector 관련  #include <vector> //vector를 사용하기 위한 헤더  #include <algorithm> //sort 알고리즘을 사용하기 위한 헤더  //Win32  #include <Windows.h> |

StudentID

|  |
| --- |
| 학생의 정보를 저장할 구조체와 vector, 좌표를 저장할 pair 객체, 박스와 정보를 출력하는데 사용될 변수들이 정의되어 있다. |
|  |

HW1\_201601607(이명진).cpp

진입점 함수인 WinMain 함수가 존재하는 파일로

# 2: 구현 주요 내용

상자 생성과 출력

마우스 좌클릭 시 좌표를 측정하는 함수인 inRect() 함수를 사용하여 사각형 안을 터치했는지를 판단한다. 이 때 pair 객체를 사용해서 x,y 좌표를 동시에 받는 방법을 사용하였다.

|  |
| --- |
| 윈도우 프로시저 내의 swtich 문 - 좌클릭 시 동작 |
| case WM\_LBUTTONDOWN:  result = inRECT(LOWORD(lParam), HIWORD(lParam));  if (result == "PUSH")  {  push = true;  InvalidateRect(hWnd, NULL, false);  }  else  {  pop = true;  std::pair<int, int> tmp = getlastPosition();  RECT rect;  rect.left = tmp.first - 1;  rect.right = tmp.first + 200 + 1;  rect.top = tmp.second - 1;  rect.bottom = tmp.second + 50 + 1;  vectorpop();  InvalidateRect(hWnd, &rect, true);  }  return 0; |
| 클릭한 좌표를 측정하는 함수 inRect() |
| std::string inRECT(int x, int y) // 사각형 안을 터치했는지 아닌지를 판정하는 함수  {  std::pair<int, int> tmp = getlastPosition();  if ((x >= tmp.first && x <= tmp.first + 200) && (y >= tmp.second && y <= tmp.second + 50))  {  return "POP";  }  return "PUSH";  } |
| 상자 안을 클릭한 경우 pop를, 아닌 경우 push를 반환한다. |

push

push를 반환받은 경우 WM\_PAINT에서 vector의 생성 과정으로 넘어가게 된다.

문제에 나온 조건인 8개 이상인 경우 더이상 박스를 생성하지 못하도록 조건문을 넣었다.

|  |
| --- |
| if (drawVector(hWnd) == -1) // -1을 리턴받을 경우 8개 이상의 박스가 존재하므로 메시지 출력  {  MessageBox(0, \_T("최대 개수 초과"), \_T("Be Advised"), MB\_OK);  } |
|  |

조건문에서 1회 drawVector() 함수가 동작하므로 박스가 생성되고 vector에 값이 생성되어 들어가게 된다.

pop

pop의 경우 가장 최근 생성된 vector의 좌표 값을 이용해 박스를 제거하고 vector의 내용을 제거하는 vectorpop() 함수를 사용해 vector의 내용을 제거하는 동작을 수행하는 동작을 수행한다.

학생의 정보는 score와 name 두 필드를 갖는 구조체로 구현되었다.  
클릭 시 학생의 이름은 생성해놓은 배열에서 임의로 값을 선택해 이름을 생성하는 rtnName() 함수로, 점수는 50부터 99사이의 임의의 정수를 반환하는 rtnScore() 함수로 생성하여 학생 정보 1개가 만들어진다.  
만들어진 정보를 student 구조체 타입의 임시 변수를 하나 생성하여 vector에 push\_back() 해주는 동작을 수행한다.

|  |
| --- |
| 학생 정보를 vector에 삽입하는 과정 |
| student tempObject = { rtnScore(), rtnName() };  vec.push\_back(tempObject);  std::string score = std::to\_string(vec.back().score); // Int형인 score를 string형으로 변환  std::string nameAndScore = vec.back().name + " " + score; // name과 score를 합쳐서 string으로 합침  std::wstring tmp = String\_Transfer(nameAndScore); //변환 함수를 사용해 string 타입을 LPCWSTR로 변환  LPCWSTR Disp = tmp.c\_str(); |
| 학생의 정보를 만드는 함수 |
| //학생의 정보를 만들기 위한 함수  std::string rtnName() //이름 생성 함수  {  std::string studentName = familyName[rand() % 12] + firstName[rand() % 12] + secondName[rand() % 12];  return studentName;  }  int rtnScore() //점수 생성 함수  {  int score = rand() % 50 + 50;  return score;  } |

또한 string 타입을 Win32API에서 사용하는 LPCWSTR 타입으로 안전하게 변환하기 위해서 String\_Transfer() 함수를 사용하여 타입 변환 작업을 수행하였다.

|  |
| --- |
| String\_Transfer() |
| std::wstring String\_Transfer(const std::string& s) // String을 LPCWSTR로 변환하는 함수  {  int len;  int slength = (int)s.length() + 1;  len = MultiByteToWideChar(CP\_ACP, 0, s.c\_str(), slength, 0, 0);  wchar\_t\* buf = new wchar\_t[len];  MultiByteToWideChar(CP\_ACP, 0, s.c\_str(), slength, buf, len);  std::wstring r(buf);  delete[] buf; //메모리 누수는 막아야 한다  return r;  } |

생성되는 박스의 좌표 값을 저장하기 위해서 pair 객체로 x,y 좌표를 저장하였고 getlastPosition() 함수와 setlastPosition() 함수를 사용하여 pair 값을 읽거나 쓰는 것이 가능하도록 작성하였다.

|  |
| --- |
| 좌표값을 저장하는 pair 객체와 함수 |
| std::pair<int, int> pos; // x,y 좌표를 저장할 pair 객체  std::pair<int, int> textpos; // 정렬을 위해 x,y 좌표를 사용할 pair 객체 |
| std::pair<int, int> getlastPosition()  {  return pos;  }  void setlastPosition(int y)  {  pos.second = y + 5;  } |

문제에 나온 vector의 값을 정렬하여 출력하는 동작을 구현하기 위해서 RewriteVector() 함수를 작성하였다.  
- 좌표값을 저장하는 pair 객체를 사용해 정렬된 값을 올바른 위치에 출력할 수 있도록 작성  
- vector의 내용을 정렬하기 위해 STL의 sort 알고리즘을 사용  
-> vector의 student 타입을 디폴트 정렬자로 정렬할 수 없으므로 compare 커스텀 비교자를 작성

|  |
| --- |
| RewriteVector() 함수 |
| void RewriteVector(HWND hwnd) // push 동작 이후 vector의 값을 정렬해 출력하는 함수  {  textpos.second = screenbottom; //가장 밑부분  for (std::vector<student>::size\_type i = 0; i < vec.size(); i++)  {  std::sort(vec.begin(), vec.end(), compare);  std::string tes = std::to\_string(i);  std::string score = std::to\_string(vec[i].score);  std::string nameAndScore = vec[i].name + " " + score + tes;  std::wstring tmp = String\_Transfer(nameAndScore);  LPCWSTR Disp = tmp.c\_str();  textpos.second = textpos.second - 55;  TextOut(ps.hdc, drawtextleft, textpos.second + 15, Disp, 6);  }  } |
| 커스텀 비교자 compare |
| bool compare(student a, student b)  {  return a.score > b.score;  } |

drawVector()와 같은 동작으로 정렬된 vector가 존재하는 박스 위에 올바르게 textout을 수행할 수 있도록 작성하였다.

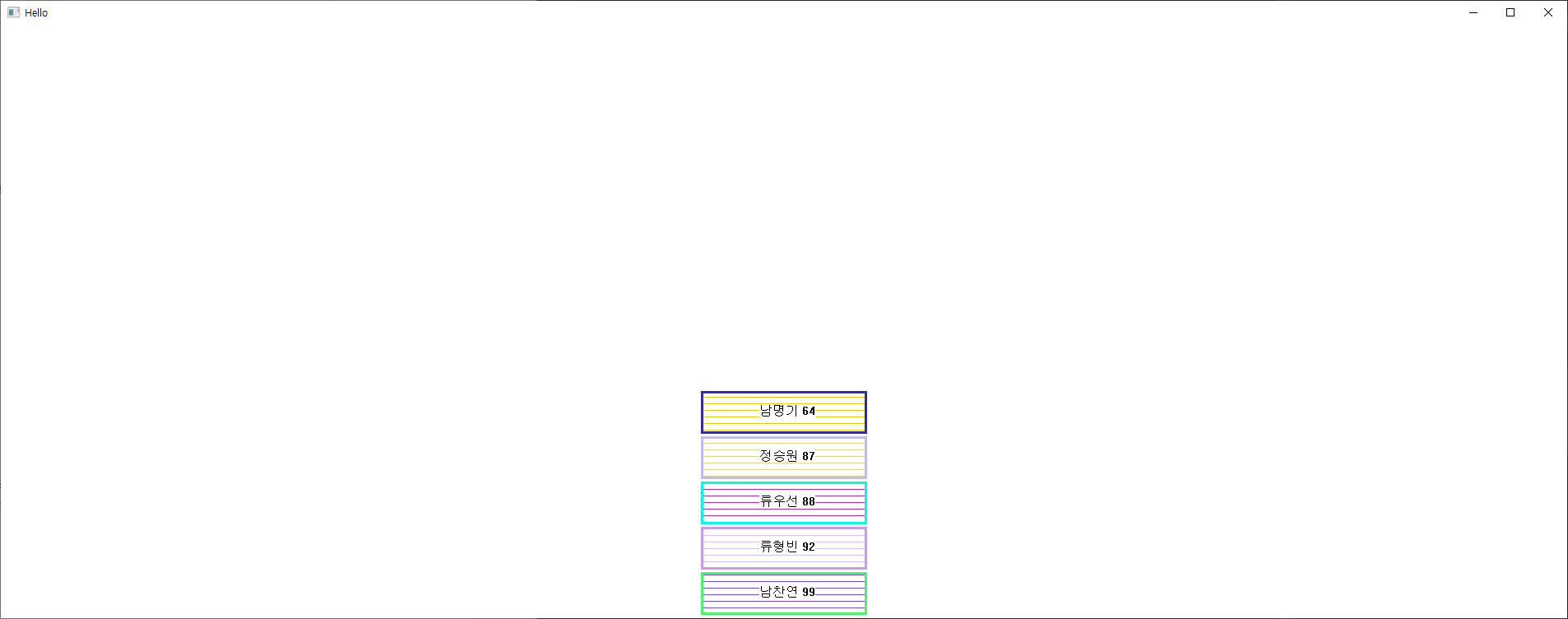
# 3: 동작 결과



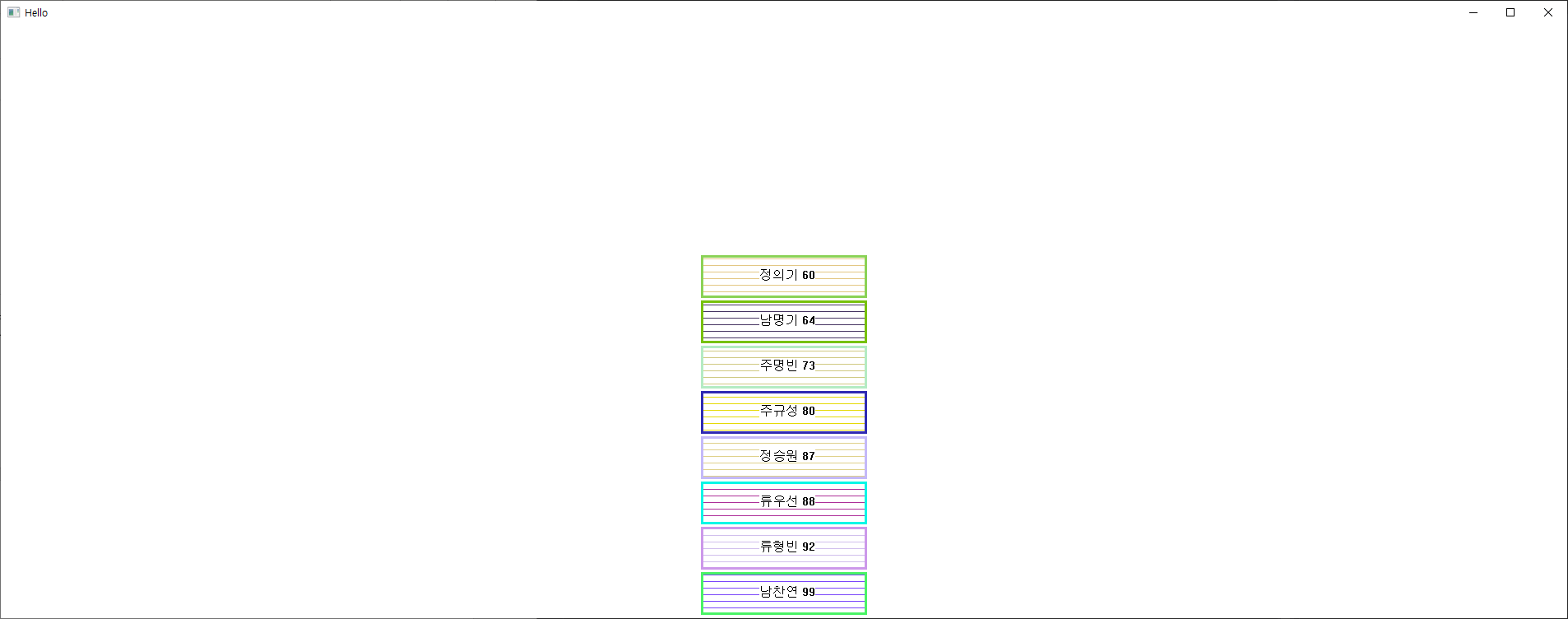
- 기본 화면으로 아무런 동작도 하지 않은 상태

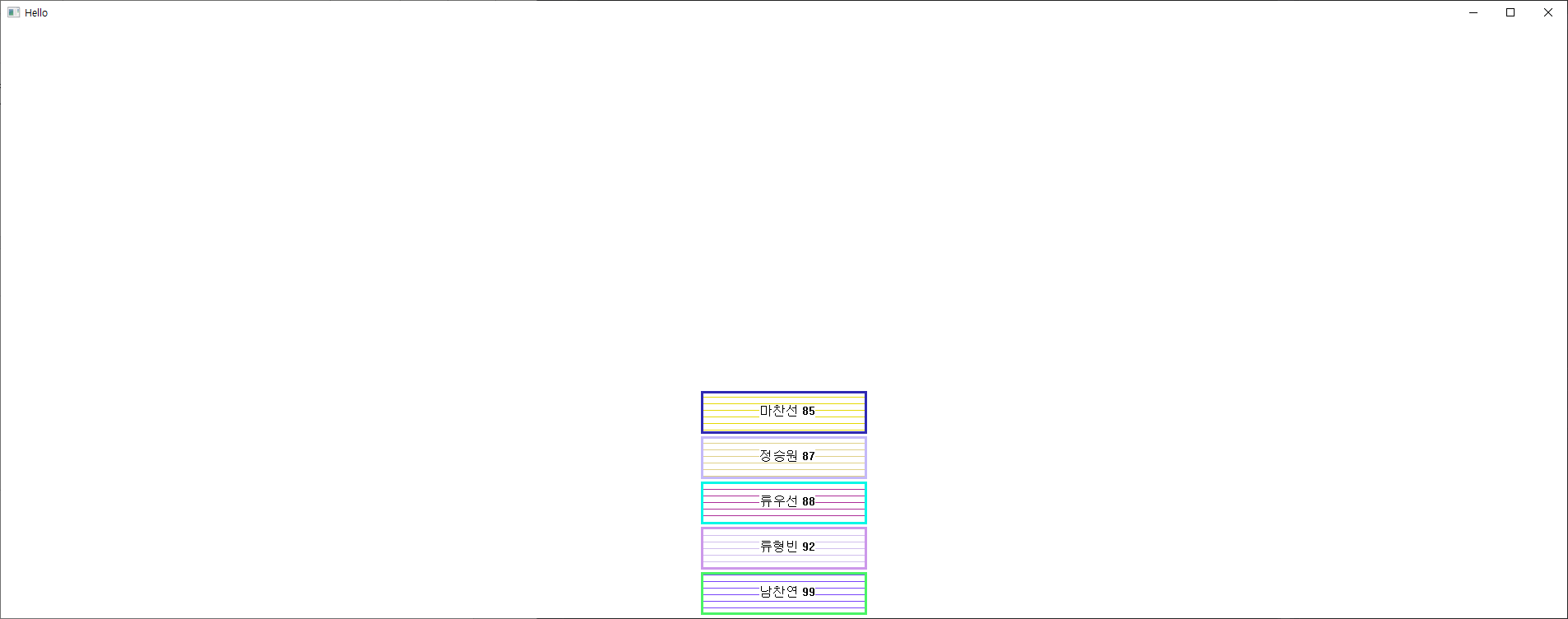


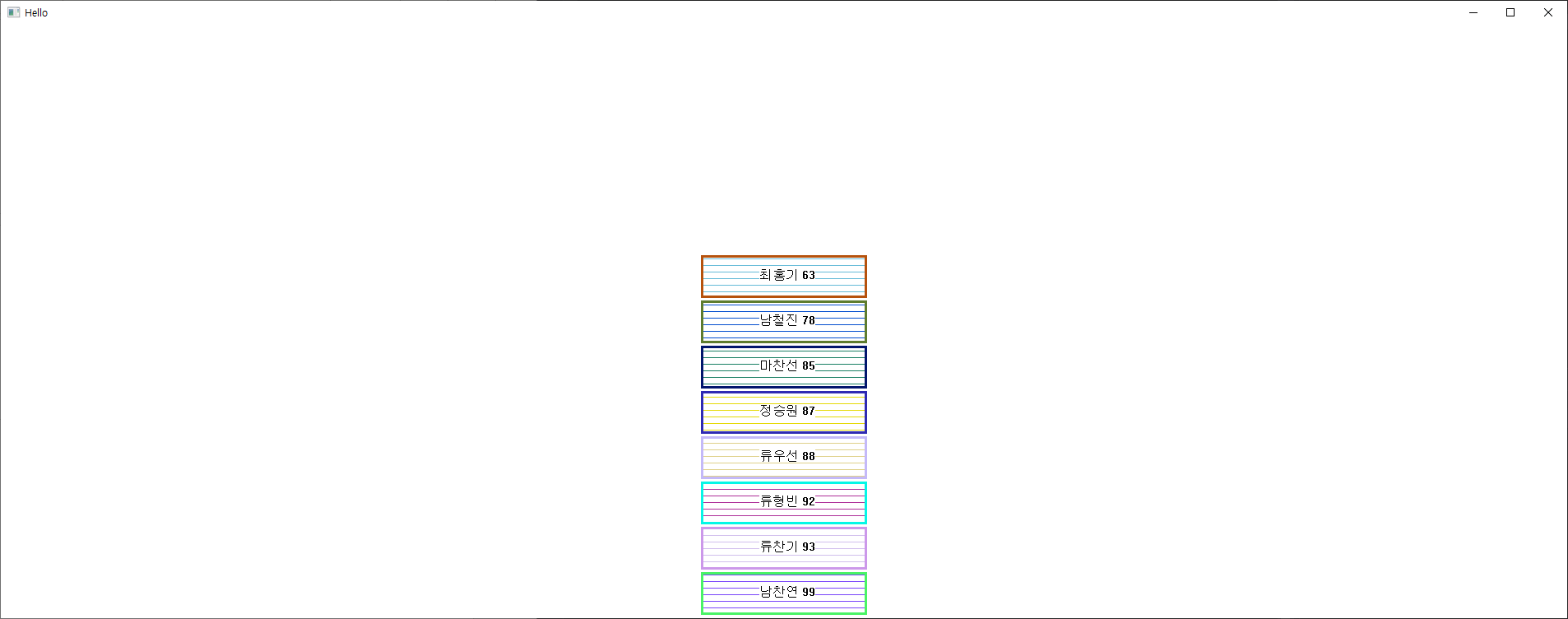
- 클릭 시 vector에 추가된 내용을 읽어와 박스를 생성한다



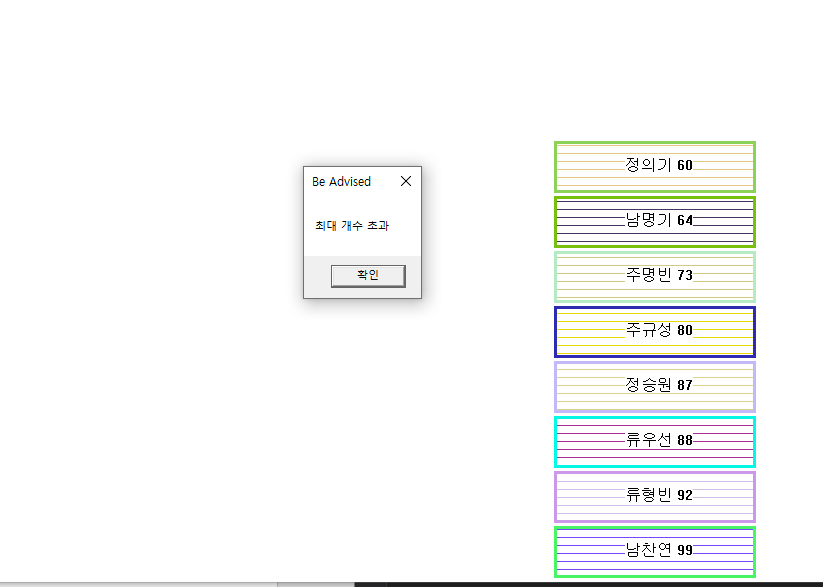
- 클릭을 반복해 박스를 쌓은 모습으로 성적이 높은 값이 아래에 배치되는 모습  
- 생성되는 박스의 색도 rand() 함수를 사용해서 각각 다르다. (랜덤이기에 같을 수도 있다)







- 정상적으로 pop, push 과정이 동작하며 정렬 기준에 맞게 정렬되는 모습



- 구현 주요 내용에서 언급한 최대 개수인 8개 초과 시 경고 문구 출력

# 4: 참고 자료

- C++와 DirectX로 배우는 2D 게임 프로그래밍 기초 – 박종승 저 (강의 교과서)

- 201501438\_박지현 학우의 힌트 코드

# 5: 전체 코드

|  |
| --- |
| stdafx.h |
| #pragma once  #include <iostream>  #include <tchar.h> //유니코드를 사용하기 위한 헤더  #include <string>  #include <ctime>;  //vector 관련  #include <vector> //vector를 사용하기 위한 헤더  #include <algorithm> //sort 알고리즘을 사용하기 위한 헤더  //Win32  #include <Windows.h> |

|  |
| --- |
| stdafx.cpp |
| #include "stdafx.h" |

|  |
| --- |
| StudentID.h |
| #pragma once  #include "stdafx.h"  // 학생 구조체 배열  struct student  {  int score;  std::string name;  };  LRESULT drawVector(HWND hwnd);  void init(HWND hwnd);  std::pair<int, int> getlastPosition();  void setlastPosition(int y);  void vectorpop();  void RewriteVector(HWND hwnd); |

|  |
| --- |
| StudentID.cpp |
| #include "StudentID.h"  #include <tchar.h>  // --------------- 선언부 --------------------  // 전역 변수  std::vector<student> vec; // 학생의 정보를 저장할 vector  std::pair<int, int> pos; // x,y 좌표를 저장할 pair 객체  std::pair<int, int> textpos; // 정렬을 위해 x,y 좌표를 사용할 pair 객체  //정보를 반환하는데 사용하는 변수들  int screenleft, screentop, screenright, screenbottom = 0;  int drawleft, drawtextleft;  PAINTSTRUCT ps;  // 학생의 이름을 만들때 사용할 배열  std::string familyName[12] =  {  "김", "이", "박", "최",  "남", "정", "한", "류",  "신", "유", "마", "주"  };  std::string firstName[12] =  {  "형", "남", "명", "승",  "학", "의", "찬", "우",  "홍", "철", "규", "대"  };  std::string secondName[12] =  {  "현", "진", "준", "성",  "기", "지", "은", "훈",  "연", "원", "빈", "선"  };  //함수  void init(HWND hwnd); //동작 함수  //vector의 내용을 생성하거나 변환해주는 함수  bool compare(student a, student b); // 커스텀 비교 함수  std::string rtnName(); //이름 생성 함수  int rtnScore(); //점수 생성 함수  std::wstring String\_Transfer(const std::string& s); // String을 LPCWSTR로 변환하는 함수  //vector 박스 관련 함수  std::pair<int, int> getlastPosition(); // vector의 마지막 위치를 얻어낼 함수  void setlastPosition(int y); // vector가 쌓일 마지막 위치를 위한 함수  void vectorpop(); // vector의 pop 동작을 위한 함수  // --------------- 선언부 --------------------  // --------------- 구현부 --------------------  void init(HWND hwnd) {  if (screenbottom != 0) // init함수가 또 한번 불리는 것을 막기 위해서이다.  {  return;  }  RECT rc;  GetClientRect(hwnd, &rc);  screenleft = rc.left;  screentop = rc.top;  screenright = rc.right;  screenbottom = rc.bottom; //screen 사이즈를 갖고와서 각각의 screen 인자에 넣어준다.  drawleft = (screenright - screenleft) / 2 - 100; // vector가 쌓일 x좌표  drawtextleft = drawleft + 100 - 30; // text가 기록될 x좌표  pos.first = drawleft; // vector에 가장 최근 vector의 x,y좌표를 넣어준다.  pos.second = screenbottom;  srand(time(NULL));  }  LRESULT drawVector(HWND hwnd) //vector를 출력하는데 사용하는 함수  {  BeginPaint(hwnd, &ps);  HPEN hMyPen = CreatePen(PS\_SOLID, 3, RGB(rand() % 255, rand() % 255, rand() % 255)); // 펜을 생성함  HGDIOBJ hOldPen = SelectObject(ps.hdc, hMyPen); // 펜을 선택함  if (vec.size() == 8) // 허용된 최대 vector의 크기는 8로 이 크기를 넘어서는 경우를 제거하기 위한 부분  {  return -1; // -1을 리턴시킨다  }  // 정상 동작한 경우 student 객체를 하나 생성하고 vector에 push\_back 한다  student tempObject = { rtnScore(), rtnName() };  vec.push\_back(tempObject);  std::string score = std::to\_string(vec.back().score); // Int형인 score를 string형으로 변환  std::string nameAndScore = vec.back().name + " " + score; // name과 score를 합쳐서 string으로 합침  std::wstring tmp = String\_Transfer(nameAndScore); //변환 함수를 사용해 string 타입을 LPCWSTR로 변환  LPCWSTR Disp = tmp.c\_str();  pos.second = pos.second - 55; // vector박스 사이에 간격을 띄움  HBRUSH MyBrush, OldBrush;  COLORREF color = RGB(rand() % 255, rand() % 255, rand() % 255);  MyBrush = CreateHatchBrush(HS\_HORIZONTAL, color);  OldBrush = (HBRUSH)SelectObject(ps.hdc, MyBrush);  Rectangle(ps.hdc, drawleft, pos.second, drawleft + 200, screentop + pos.second + 50);  TextOut(ps.hdc, drawtextleft, pos.second + 15, Disp, 6);  return 0;  }  void RewriteVector(HWND hwnd) // push 동작 이후 vector의 값을 정렬해 출력하는 함수  {  textpos.second = screenbottom; //가장 밑부분  for (std::vector<student>::size\_type i = 0; i < vec.size(); i++)  {  std::sort(vec.begin(), vec.end(), compare);  std::string tes = std::to\_string(i);  std::string score = std::to\_string(vec[i].score);  std::string nameAndScore = vec[i].name + " " + score + tes;  std::wstring tmp = String\_Transfer(nameAndScore);  LPCWSTR Disp = tmp.c\_str();  textpos.second = textpos.second - 55;  TextOut(ps.hdc, drawtextleft, textpos.second + 15, Disp, 6);  }  }  std::pair<int, int> getlastPosition()  {  return pos;  }  void setlastPosition(int y)  {  pos.second = y + 5;  }  void vectorpop()  {  if (vec.size() >= 1)  {  vec.pop\_back();  }  }  bool compare(student a, student b)  {  return a.score > b.score;  }  // 학생의 정보를 만들기 위한 함수  std::string rtnName() //이름 생성 함수  {  std::string studentName = familyName[rand() % 12] + firstName[rand() % 12] + secondName[rand() % 12];  return studentName;  }  int rtnScore() //점수 생성 함수  {  int score = rand() % 50 + 50;  return score;  }  std::wstring String\_Transfer(const std::string& s) // String을 LPCWSTR로 변환하는 함수  {  int len;  int slength = (int)s.length() + 1;  len = MultiByteToWideChar(CP\_ACP, 0, s.c\_str(), slength, 0, 0);  wchar\_t\* buf = new wchar\_t[len];  MultiByteToWideChar(CP\_ACP, 0, s.c\_str(), slength, buf, len);  std::wstring r(buf);  delete[] buf; //메모리 누수는 막아야 한다  return r;  }  // --------------- 구현부 -------------------- |

|  |
| --- |
| HW1\_201601607(이명진) – 진입점 |
| #include "stdafx.h" //미리 컴파일된 헤더  #include "StudentID.h" //StudentID의 헤더  // --------------- 선언부 --------------------  // 전역 변수  HWND hWndMain = 0;  bool push = false;  bool pop = false;  std::string result;  //함수  bool initWin(HINSTANCE hInstance, int nCmdShow);  int runWin();  LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam);  int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR pCmdLine, int nShowCmd);  bool initWin(HINSTANCE hInstance, int nCmdShow);  std::string inRECT(int x, int y); // 사각형 안을 터치했는지 아닌지를 판정하는 함수  // --------------- 선언부 --------------------  // --------------- 구현부 --------------------  int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR pCmdLine, int nShowCmd) //진입점 함수: WinMain  {  if (!initWin(hInstance, nShowCmd))  {  MessageBox(0, \_T("initWin - failed"), \_T("Error"), MB\_OK);  return 0;  }  return runWin();  }  bool initWin(HINSTANCE hInstance, int nCmdShow) //윈도우 생성 함수  {  WNDCLASS wc;  wc.style = CS\_HREDRAW | CS\_VREDRAW;  wc.lpfnWndProc = (WNDPROC)WndProc;  wc.cbClsExtra = 0;  wc.cbWndExtra = 0;  wc.hInstance = hInstance;  wc.hIcon = LoadIcon(hInstance, (LPCTSTR)IDI\_APPLICATION);  wc.hCursor = LoadCursor(NULL, IDC\_ARROW);  wc.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);  wc.lpszMenuName = 0;  wc.lpszClassName = \_T("Hello");  if (!RegisterClass(&wc))  {  MessageBox(0, \_T("RegisterClass - failed"), 0, 0);  return false;  }  hWndMain = CreateWindow(\_T("Hello"), \_T("Hello"), WS\_OVERLAPPEDWINDOW,  CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT,  NULL, NULL, hInstance, NULL);  if (!hWndMain)  {  MessageBox(0, \_T("CreateWindow - failed"), 0, 0);  return false;  }  ShowWindow(hWndMain, nCmdShow);  UpdateWindow(hWndMain);  return true;  }  int runWin() //메시지 루프 함수  {  MSG msg;  ZeroMemory(&msg, sizeof(MSG));  while (GetMessage(&msg, NULL, 0, 0) != 0)  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  return (int)msg.wParam;  }  LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam) //윈도우 프로시저  {  init(hWnd);  switch (msg)  {  case WM\_LBUTTONDOWN:  result = inRECT(LOWORD(lParam), HIWORD(lParam));  if (result == "PUSH")  {  push = true;  InvalidateRect(hWnd, NULL, false);  }  else  {  pop = true;  std::pair<int, int> tmp = getlastPosition();  RECT rect;  rect.left = tmp.first - 1;  rect.right = tmp.first + 200 + 1;  rect.top = tmp.second - 1;  rect.bottom = tmp.second + 50 + 1;  vectorpop();  InvalidateRect(hWnd, &rect, true);  }  return 0;  case WM\_KEYDOWN:  if (wParam == VK\_ESCAPE)  {  DestroyWindow(hWndMain);  }  return 0;  case WM\_DESTROY:  PostQuitMessage(0);  return 0;  case WM\_PAINT:  if (push == true) // push 동작  {  if (drawVector(hWnd) == -1) // -1을 리턴받을 경우 8개 이상의 박스가 존재하므로 메시지 출력  {  MessageBox(0, \_T("최대 개수 초과"), \_T("Be Advised"), MB\_OK);  }  else //아닌 경우 vector의 값을 갱신한다  {  RewriteVector(hWnd);  }  }  else if (pop == true) //pop 동작, 별도의 갱신 동작이 불필요하다  {  setlastPosition(getlastPosition().second + 50);  }  //사용한 값을 리셋  push = false;  pop = false;  break;  default:  return DefWindowProc(hWnd, msg, wParam, lParam);  }  return 0;  }  std::string inRECT(int x, int y) // 사각형 안을 터치했는지 아닌지를 판정하는 함수  {  std::pair<int, int> tmp = getlastPosition();  if ((x >= tmp.first && x <= tmp.first + 200) && (y >= tmp.second && y <= tmp.second + 50))  {  return "POP";  }  return "PUSH";  } |