|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2019\_1\_C++ \_05 | 학번 : |  | 이름 : |  |

* **강의 내용**

|  |
| --- |
| 1. **기본 타입 변수에 대한 참조** |
| int main() {  cout << "i" << '\t' << "n" << '\t' << "refn" << endl;  int i = 1;  int n = 2;  **int &refn = n; // 참조 변수 refn 선언. refn은 n에 대한 별명**  n = 4;  refn++; **// refn=5, n=5**  cout << i << '\t' << n << '\t' << refn << endl;  refn = i; **// refn=1, n=1**  refn++; **// refn=2, n=2**  cout << i << '\t' << n << '\t' << refn << endl;    **int \*p = &refn; // p는 n의 주소를 가짐, 참조에대한 포인터 변수 선언**  \*p = 20; **// refn=20, n=20**  cout << i << '\t' << n << '\t' << refn << endl;  } |
| **[실행결과]** |

|  |
| --- |
| 1. 참조에 의한 호출 – 객체 참조 전달 |
| class Circle {  private:  int radius;  public:  Circle();  Circle(int r);  ~Circle();  double getArea() { return 3.14\*radius\*radius; }  int getRadius() { return radius; }  void setRadius(int radius) { this->radius = radius; }  };  Circle::Circle() {  radius = 1;  cout << "생성자 실행 radius = " << radius << endl;  }  Circle::Circle(int radius) {  this->radius = radius;  cout << "생성자 실행 radius = " << radius << endl;  }  Circle::~Circle() {  cout << "소멸자 실행 radius = " << radius << endl;  }  void increaseCircle(Circle &c) {  int r = c.getRadius();  c.setRadius(r+1);  }  int main() {  Circle waffle(30);  increaseCircle(waffle); //참조에 의한 호출  cout << waffle.getRadius() << endl;  } |
| **[실행결과]** |

|  |
| --- |
| 1. 참조 리턴 |
| #include <iostream>  using namespace std;  // 배열 s의 index 원소 공간에 대한 참조 리턴하는 함수  char& find(char s[], int index) {  return s[index]; // 참조 리턴  }  int main() {  char name[] = "Mike";  cout << name << endl;  find(name, 0) = 'S'; // name[0]='S'로 변경  cout << name << endl;  char& ref = find(name, 2);  ref = 't'; // name = "Site"  cout << name << endl;  } |
| **[실행결과]** |

|  |
| --- |
| 1. 얕은 복사 - 문제 발생 |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <iostream>  #include <cstring>  using namespace std;  class Person { // Person 클래스 선언  char\* name;  int id;  public:  Person(int id, const char\* name); // 생성자  ~Person(); // 소멸자  void changeName(const char \*name);  void show() { cout << id << ',' << name << endl; }  };  Person::Person(int id, const char\* name) { // 생성자  this->id = id;  int len = strlen(name); // name의 문자 개수  this->name = new char[len + 1]; // name 문자열 공간 핟당  strcpy(this->name, name); // name에 문자열 복사  }  Person::~Person() {// 소멸자  if (name) // 만일 name에 동적 할당된 배열이 있으면  delete[] name; // 동적 할당 메모리 소멸  }  void Person::changeName(const char\* name) { // 이름 변경  if (strlen(name) > strlen(this->name))  return;  strcpy(this->name, name);  }  int main() {  Person father(1, "Kitae"); // (1) father 객체 생성  Person daughter(father); // (2) daughter 객체 복사 생성. 복사생성자호출  cout << "daughter 객체 생성 직후 ----" << endl;  father.show(); // (3) father 객체 출력  daughter.show(); // (3) daughter 객체 출력  daughter.changeName("Grace"); // (4) daughter의 이름을 "Grace"로 변경  cout << "daughter 이름을 Grace로 변경한 후 ----" << endl;  father.show(); // (5) father 객체 출력  daughter.show(); // (5) daughter 객체 출력  return 0; // (6), (7) daughter, father 객체 소멸  } |
| **[실행결과]** |

|  |
| --- |
| 1. 깊은 복사 |
| **//4)번 예제에 아래의 코드를 추가하여 결과 확인**  **Person::Person(Person& person) { // 복사 생성자**  **this->id = person.id; // id 값 복사**  **int len = strlen(person.name);// name의 문자 개수**  **this->name = new char[len + 1]; // name을 위한 공간 핟당**  **strcpy(this->name, person.name); // name의 문자열 복사**  **cout << "복사 생성자 실행. 원본 객체의 이름 " << this->name << endl;**  **}** |
| **[실행결과]** |

* 내용 점검 – 5장 연습문제

|  |  |
| --- | --- |
| 번호 | 문제풀이 |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 | (1)  (2) |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 | (1)  (2)  (3)  (4) |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 | (1)  (2)  (3) |
| 16 |  |
| 17 |  |
| 18 |  |

* 프로그램 과제

|  |
| --- |
| 1. 교재 270p 문제3번 |
| **[프로그램 소스]** |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 270p 문제5번 |
| **[프로그램 소스]** |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 271p 문제6번 |
| **[프로그램 소스]** |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 272p 문제8번 |
| **[프로그램 소스]** |
| **[실행결과]** |