|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2019\_1\_C++ \_05 | 학번 : | 20145128 | 이름 : | 박인근 |

* **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 | 4 |
| 2 | 1 |
| 3 | 주소 |
| 4 | 1. 같다 (2) 다르다 |
| 5 | (1) 5  (2) 25 |
| 6 | 1 4 9 |
| 7 | 2 |
| 8 | 2 |
| 9 | 1 |
| 10 | (1) array [] = { 0, 2, 4, 6, 8, 10, 12, 14, 16, 100 } // f(9) 값이 100 으로 바뀜  (2) array [] = { 0, 4, 6, 8, 10, 12, 14, 16, 18, 18 } // f(1)~f(8) 까지의 값들을 2씩 증가  (3) array [] = { 0, 2, 4, 6, 8, 10, 12, 14, 16, 18 } // 배열은 변함 없음  (4) array [] = { 0, 2, 4, 6, 0, 10, 12, 14, 16, 18 } // // f(2) = 4 , 이며 f(4) = 0 이기 때문에 f(4)의 위치에 0이 삽입 |
| 11 | 함수가 값에 의한 호출로 a, b 값에는 실제로 영향을 줄 수가 없음. 참조에 의한 호출로 바꿔줘야 함.  void copy(int& dest, int src){  dest = src;  } |
| 12 | big1 함수에서는 a,b 와 x,y 가 각각 다른 공간을 사용하므로 big1 함수 실행시 x = 1, y =2 그대로이고 big2 함수에서는 a,b 가 참조변수이므로 x,y 값과 a,b 값이 같아진다. 따라서 big2 함수를 실행시 x와 비교했을 때 y가 큰값이며 w=100 을 넣었기 때문에 x = 1 , y = 100이 된다. |
| 13 | 기본 생성자 : MyClass(); 복사 생성자 : MyClass(MyClass&); |
| 14 | 2 |
| 15 | (1) MyClass::~MyClass() { delete element; }  (2) MyClass::MyClass(MyClass& a) {  size = a.size;  element = a.element;  }  (3) MyClass::MyClass(MyClass& a) {  size = a.size;  element = new int[size];  for(int i = 0 ; i < size ; i ++) element[i] = a.element[i];  } |
| 16 | 1 |
| 17 | Student::Student(Student& a){  name = a.name;  id = a.id;  grade = a.grade;  } |
| 18 | Student::Student(Student& a){  pName = a.pName;  pId = a.pId;  grade = a.grade; |

* 프로그램 과제

|  |
| --- |
| 1. 교재 270p 문제3번 |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  void combine(string& t1, string &t2, string &t3) {  t3 = t1 + " " + t2;  }  int main() {  string text1("I love you"), text2("very much");  string text3; //비어 있는 문자열  combine(text1, text2, text3); // text1과 " ", 그리고 text2를 덧붙여 text3 만들기  cout << text3; // "I love you very much" 출력  cout << endl;  system("pause");  } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 270p 문제5번 |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  class Circle {  int radius;  public:  Circle(int r) { radius = r; }  int getRadius() { return radius; }  void setRadius(int r) { radius = r; }  void show() { cout << "반지름이 " << radius << "인 원" << endl; }  };  void increaseBy(Circle& a, Circle& b) {  int r = a.getRadius() + b.getRadius();  a.setRadius(r);  }  int main() {  Circle x(10), y(5);  increaseBy(x, y); //x의 반지름이 15인 원을 만들고자 한다.  x.show(); // "반지름이 15인 원"을 출력한다.  system("pause");  } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 271p 문제6번 |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  char& find(char a[], char c, bool& success) {  for (int i = 0; i < strlen(a); i++) {  if (a[i] == c) {  success = true;  return a[i];  }  }  success = false;  }  int main() {  char s[] = "Mike";  bool b = false;  char& loc = find(s, 'M', b);  if (b == false) {  cout << "M을 발견할 수 없다" << endl;  return 0;  }  loc = 'm'; // 'M' 위치에 'm' 기록  cout << s << endl; // "mike"가 출력됨  system("pause");  } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 272p 문제8번 |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  class MyIntStack {  int \*p;  int size;  int tos;  public:  MyIntStack();  MyIntStack(int size);  MyIntStack(MyIntStack& s); //복사 생성자  ~MyIntStack();  bool push(int n); //정수 n을 스택에 푸시한다.  //스택이 꽉 차 이쓰면 false를, 아니면 true 리턴  bool pop(int &n); //스택의 탑에 있는 값을 n에 팝한다.  // 만일 스택이 비어있으면 false를, 아니면 true 리턴  };  MyIntStack::MyIntStack(){}  MyIntStack::MyIntStack(int size) {  this->size = size;  p = new int[size];  tos = -1; // 스택의 탑을 가르키는 인덱스로서 -1 부터 시작을 해야 스택이 비어있다는 것을 나타낼 수 있음.  }  MyIntStack::MyIntStack(MyIntStack& s) {  size = s.size;  tos = s.tos;  p = new int[size];  for (int i = 0; i < size; i++) p[i] = s.p[i];  }  MyIntStack::~MyIntStack() {  delete p;  }  bool MyIntStack::push(int n) {  tos++; //스택에 push 했으므로 인덱스 1 증가  if (tos >= 10) return false; // 스택 꽉 참  else {  p[tos] = n; //스택의 탑 원소에 n 삽입  return true; // 스택이 꽉 차지 않음  }  }  bool MyIntStack::pop(int &n) {  if (tos < 0) // 스택이 비어있음  return false;  else {  n = p[tos];  return true;  }  tos--; //스택에 pop 했으므로 인덱스 1 감소  }  int main() {  MyIntStack a(10);  a.push(10);  a.push(20);  MyIntStack b = a; //복사 생성  b.push(30);  int n;  a.pop(n); // 스택 a 팝  cout << "스택 a에서 팝한 값 " << n << endl;  b.pop(n); // 스택 b 팝  cout << "스택 b에서 팝한 값 " << n << endl;  system("pause");  } |
| **[실행결과]** |