|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2019\_1\_C++ \_05 | 학번 : | 20145165 | 이름 : | 정균모 |

* **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 | Call by address |
| 4 | 1. 같다 (2)같다 |
| 5 | (1)5  (2)25 |
| 6 | 1 4 9 |
| 7 | 2 |
| 8 | 2 |
| 9 | 1 |
| 10 | (1) 0, 2, 4, 6, 8, 10, 12, 14, 100, 18  (2) 0, 4, 6, 8, 10, 12, 14, 16, 18, 20  (3) 0, 2, 4, 6, 8, 10, 12, 14, 16, 18  (4) 0, 2, 4, 6, 0, 10, 12, 14, 100, 18 |
| 11 | 복사되지 않는 이유는 main에 a,b의 메모리 주소와 함수에 있는 dest, src의 메모리 주소가 다르기 때문입니다. Dest 가 src로 값을 받는다고 해도 함수 내에서만 적용이 됩니다. 이를 해결하기위해 call by valus가 아닌 call by address로 만들면 됩니다.  void copy(int &dest, int &src) {  dest = src;  } |
| 12 | z에게 100을 주었을 때는 x,y 값은 변하지 않지만 w에게 100을 주었을 때는 y값이 100이 된다. Big2가 x,y의 주소 값을 받아서 w에게 y의 주소값을 주었기 때문이다. |
| 13 | 기본 생성자: MyClass::MyClass() { }  복사 생성자: MyClass (const MyClass &); |
| 14 | 2 |
| 15 | (1) ~MyClass() { delete[] element; }  (2) MyClass (const MyClass &);  (3) MyClass(const MyClass& m) {  size = m.size;  element = new int[size];  for (int i = 0; i < size; i++) element[i] = m.element[i];  } |
| 16 | 4 |
| 17 | Student(const Student& s) {  name = s.name;  id = s.id;  Grade = s.Grade;  } |
| 18 | Student(string \*pname, string \*pid, double grade) {  pname = new string[pname->length + 1];  this->pname = pname;  pid = new string[pid->length + 1];  this->pid = pid;  this->grade = grade;  } |

* 프로그램 과제

|  |
| --- |
| 1. 교재 270p 문제3번 |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  void combine(string& text1, string& text2, string& text3) {  text3 = text1 + text2;  }  int main() {  string text1("I love you ") , text2("very much");  string text3;  combine(text1, text2, text3);  cout << text3;  getchar();  } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 270p 문제5번 |
| **[프로그램 소스]**  #include <iostream>  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);  x.show();  system("pause");  } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 271p 문제6번 |
| **[프로그램 소스]**  #include <iostream>  using namespace std;  char& find(char a[], char c, bool& success) {  for (int i = 0; i < sizeof(a); i++)  if (a[i] == c) {  success = true;  return a[i];  }  return \*a;  }  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';  cout << s << endl;  } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 272p 문제8번 |
| **[프로그램 소스]**  #include <iostream>  using namespace std;  class MyIntStack {  int \*p;  int size;  int tos;  public:  MyIntStack();  MyIntStack(int size);  MyIntStack(MyIntStack& s);  ~MyIntStack() { delete[] p; }  bool push(int n);  bool pop(int &n);  };  MyIntStack::MyIntStack(int size) {  tos = -1;  p = new int[size];  this->size = size;  }  bool MyIntStack::push(int n) {  tos++;  if (tos == size)  return false;  else {  p[tos] = n;  return true;  }  }  bool MyIntStack::pop(int &n) {  if (tos == -1)  return false;  else {  n = p[tos];  return true;  }  tos--;  }  MyIntStack::MyIntStack(MyIntStack& s) {  this->p = new int[s.size];  this->size = s.size;  this->tos = s.tos;  for (int i = 0; i <= s.tos; i++)  this->p[i] = s.p[i];  }  int main() {  MyIntStack a(10);  a.push(10);  a.push(20);  MyIntStack b = a;  b.push(30);  int n;  a.pop(n);  cout << "스택 a에서 팝한 값 " << n << endl;  b.pop(n);  cout << "스택 b에서 팝한 값 " << n << endl;  system("pause");  } |
| **[실행결과]** |