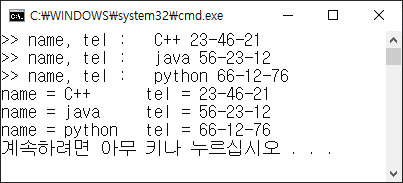
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2020\_2\_C++ \_06 | 학번 : | 20175105 | 이름 : | 곽영주 |

* **강의 내용**

1. 3개의 객체 주소를 저장할 수 있는 객체 포인터 배열을 선언하여 Person 클래스 객체 처리

class Person {

string name;

string tel;

public:

Person(string n, string t) :name(n), tel(t) { };

string getName() { return name; }

string getTel() { return tel; }

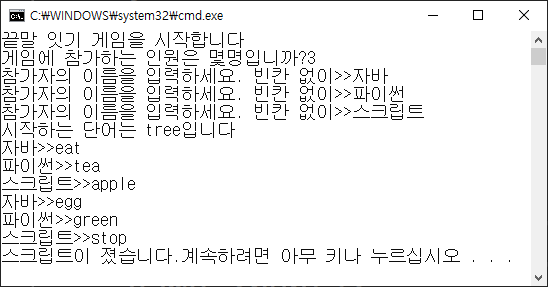
void disPlay() { cout << "name = " << name << "\ttel = " << tel << endl; }

};

|  |
| --- |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  class Person {  string name;  string tel;  public:  Person(string n, string t) : name(n), tel(t) {};  string getName() { return name; }  string getTel() { return tel; }  void disPlay() { cout << "name = " << name << "\ttel = " << tel << endl; }  };  int main() {  Person\* p[3];  string name, tel;  int size = sizeof(p) / sizeof(p[0]);  for (int i = 0; i < size; i++) {  cout << "name, tel : ";  cin >> name >> tel;  p[i] = new Person(name, tel);  }  for (int i = 0; i < size; i++) {  p[i]->disPlay();  }  for (int i = 0; i < size; i++) {  delete p[i];  }  return 0;  } |
| **[실행 결과]** |

* **프로그램 과제**

1. 제시된 결과처럼 끝말잇기 게임을 실행하는 프로그램을 작성하시오. 단, 끝말 잇기 게임에 참여하는 선수의 수를 입력 받고 선수 각 사람의 이름을 입력 받은 후 게임을 시작한다



|  |
| --- |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  class User {  string name; // 참여자 이름    public :  User() {}  void setName(string n) { name = n; }  string getName() { return name; }  bool check(string t, string w);  };  bool check(string t, string w) { // 끝말 잇기 확인  if (t[t.length() - 1] == w[0]) {  return true;  }  else {  return false;  }  }  int main() {  int size;  cout << "끝말 잇기 게임을 시작합니다" << endl;  cout << "게임에 참가하는 인원은 몇 명입니까? ";  cin >> size;  User \*u = new User[size];  string name;  for (int i = 0; i < size; i++) { // 참가자 이름 생성  cout << "참가자의 이름을 입력하세요. 빈칸 없이>> ";  cin >> name;  u[i].setName(name);  }  bool b; // 끝말 잇기 확인하는 변수  int count = 0; // 참가자 카운트  string word, temp = "tree";  cout << "시작하는 단어는 " << temp << "입니다." << endl;  while (1) {  cout << u[count].getName() << ">> ";  cin >> word;  b = check(temp, word); // 끝말 잇기 확인  if (b == true) { // 끝말 잇기 성공  if (count == (size - 1)) { // 마지막 참여자이면  count = 0; // 다시 첫번째 참여자로  }  else {  count++; // 다음 참여자로  }  temp = word; // 시작 단어 바꾸기  }  else { // 끝말 잇기 실패  break;  }  }  cout << u[count].getName() << "이(가) 졌습니다." << endl;  delete[] u;  return 0;  } |
| **[실행 결과]** |

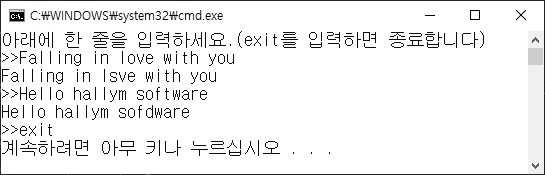
1. string 클래스를 이용하여 사용자가 입력한 영문 한 줄을 입력 받고 글자 하나만 랜덤하게 수정하여 출력하는 프로그램을 작성하시오.

힌트) #include <ctime>

#include <cstdlib>

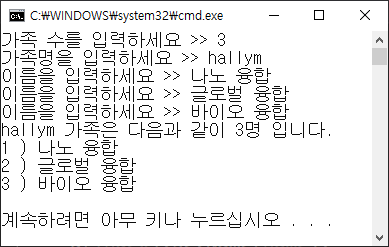
srand((unsigned)time(0));

cout<<rand(); //0에서 32767사이의 랜덤함 정수 발생



|  |
| --- |
| **[프로그램 소스]**  #include <iostream>  #include <string>  #include <ctime>  #include <cstdlib>  using namespace std;  int main() {  string s;  int index, n;  cout << "아래에 한 줄을 입력하세요.(exit를 입력하면 종료합니다)" << endl;  srand((unsigned)time(0));  while (1) {  cout << ">> ";  getline(cin, s, '\n');  if (s == "exit") { // 종료  break;  }  while (1) { // 문자열 랜덤 index생성 및 공백 문자 검사  index = rand() % s.length(); // 입력한 문자열 index 랜덤  if (s[index] != ' ') // 공백 문자 제외  break;  }  n = rand() % 2; // 대문자(0), 소문자(1) 랜덤  if(n == 0)  s[index] = (char)((rand() % 26) + 65); // 대문자 랜덤 생성  if(n == 1)  s[index] = (char)((rand() % 26) + 97); // 소문자 랜덤 생성  cout << s << endl;  }  return 0;  } |
| **[실행 결과]** |

1. Person, Family 클래스에 필요한 멤버를 추가하여 제시된 결과처럼 실행되도록 프로그램을 완성 하시오



class Person {

string name;

public:

Person(string name) { this->name = name; }

string getName() { return name; }

};

class Family {

Person\* p; // Person 배열 포인터

int size; // Person 배열의 크기. 가족 구성원 수

public:

Family(string name, int size); // size 개수만큼 Person 배열 동적 생성

void show(); // 모든 가족 구성원 출력

~Family();

};

int main() {

Family \*simpson;

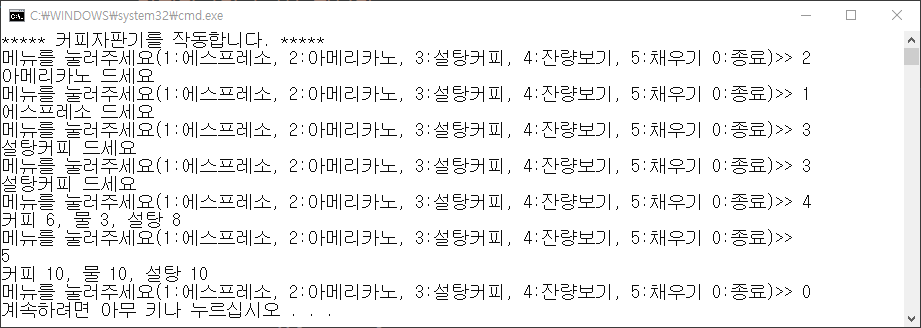
//본인 작성

delete simpson;

}

|  |
| --- |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  class Person {  string name;  public :  Person() {};  Person(string name) { this->name = name; }  string getName() { return name; }  };  class Family {  Person \*p;  int size; // 가족 구성원 수  string Family\_name; // 가족명  public :  Family(string name, int size);  void show();  ~Family();  };  Family::Family(string Family\_name, int size) {  this->Family\_name = Family\_name;  this->size = size;  p = new Person[size];  string name; // 이름    for (int i = 0; i < size; i++) {  cout << "이름을 입력하세요 >> ";  getline(cin, name);  p[i] = Person(name);  }  }  void Family::show() { // 모든 가족 구성원 출력  cout << Family\_name << " 가족은 다음과 같이 " << size << "명 입니다." << endl;  for (int i = 0; i < size; i++)  cout << i + 1 << " ) " << p[i].getName() << endl;  }  Family::~Family() {  delete[] p;  }  int main() {  Family \*simpson;  int size; // 가족 구성원 수  string Family\_name; // 가족명  cout << "가족 수를 입력하세요 >> ";  cin >> size;    cout << "가족명을 입력하세요 >> ";  cin.ignore(1, '\n');  getline(cin, Family\_name);    simpson = new Family(Family\_name, size);  simpson->show();  simpson->~Family();  delete simpson;  return 0;  } |
| **[실행 결과]** |

1. 제시된 결과처럼 실행될 수 있도록 main()함수와 CoffeeVMachine, Container를 완성하시오. 만일 커피, 물, 설탕 중 잔량이 하나라도 부족하면 “원료가 부족합니다”를 출력한다



class Container { // 통 하나를 나타내는 클래스

int size; // 현재 저장 량, 최대 저장 가능량은 10

public:

Container() { size = 10; }

void fill(int n); // n 만큼 채우기

bool consume(int n); // n 만큼 소모하기

int getSize(); // 현재 크기 리턴

};

class CoffeeVMachine { // 커피 자판기를 표현하는 클래스

Container tong[3]; // tong[0]는 커피, tong[1]은 물, tong[2]는 설탕통을 나타냄

void fill(); // 3개의 통을 모두 10으로 채움

void selectEspresso(); // 에스프레소를 선택한 경우, 커피 1, 물 1 소모

void selectAmericano(); // 아메리카노를 선택한 경우, 커피 1, 물 2 소모

void selectSugarCoffee(); // 설탕커피를 선택한 경우, 커피 1, 물 2 소모, 설탕 1 소모

void show(); // 현재 커피, 물, 설탕의 잔량 출력

bool checkInputError(); // 오류 처리를 위해 추가한 멤버 함수

public:

void run(); // 커피 자판기 작동

};

int main() {

CoffeeVMachine cvm;

cvm.run();

}

|  |
| --- |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  class Container {  int size; // 현재 저장 량 (최대 저장 가능량은 10)  public :  Container() { size = 10; }  void fill(int n);  bool consume(int n);  int getSize();  };  void Container::fill(int n) { // n만큼 채우기  size = n;  }  bool Container::consume(int n) { // n만큼 소모  if (size >= n && size != 0) {  size -= n;  return true;  }  else  return false;  }  int Container::getSize() { // 현재 저장 량 리턴  return size;  }  class CoffeeVMachine {  Container tong[3]; // [0]커피, [1]물, [2]설탕  void fill();  void selectEspresso();  void selectAmericano();  void selectSugarCoffee();  void show();  bool checkInputError(int n);  public :  void run();  };  void CoffeeVMachine::fill() { // 모두 10으로 채움  for (int i = 0; i < 3; i++)  tong[i].fill(10);  }  void CoffeeVMachine::selectEspresso() { // 에스프레소(커피 1, 물 1)  bool c = tong[0].consume(1);  bool w = tong[1].consume(1);  if (c == true and w == true)  cout << "에스프레소 드세요." << endl;  else {  // 다른 곳이 원료부족인데 딴 곳이 true로 잔량 빠진거 채우기  if (c == true)  tong[0].fill(tong[0].getSize() + 1);  if (w == true)  tong[1].fill(tong[1].getSize() + 1);  cout << "원료가 부족합니다." << endl;  }  }  void CoffeeVMachine::selectAmericano() { // 아메리카노(커피 1, 물 2)  bool c = tong[0].consume(1);  bool w = tong[1].consume(2);  if (c == true and w == true)  cout << "아메리카노 드세요." << endl;  else {  // 다른 곳이 원료부족인데 딴 곳이 true로 잔량 빠진거 채우기  if (c == true)  tong[0].fill(tong[0].getSize() + 1);  if (w == true)  tong[1].fill(tong[1].getSize() + 2);  cout << "원료가 부족합니다." << endl;  }  }  void CoffeeVMachine::selectSugarCoffee() { // 설탕커피(커피 1, 물 2, 설탕 1)  bool c = tong[0].consume(1);  bool w = tong[1].consume(2);  bool s = tong[2].consume(1);  if (c == true && w == true && s == true)  cout << "설탕커피 드세요." << endl;  else {  // 다른 곳이 원료부족인데 딴 곳이 true로 잔량 빠진거 채우기  if (c == true)  tong[0].fill(tong[0].getSize() + 1);  if (w == true)  tong[1].fill(tong[1].getSize() + 2);  if (s == true)  tong[2].fill(tong[2].getSize() + 1);  cout << "원료가 부족합니다." << endl;  }  }  void CoffeeVMachine::show() { // 현재 잔량 출력  cout << "커피 " << tong[0].getSize() << ", "  << "물 " << tong[1].getSize() << ", "  << "설탕 " << tong[2].getSize() << endl;  }  bool CoffeeVMachine::checkInputError(int n) { // 오류 처리  if (n >= 0 && n <= 5)  return true;  return false;  }  void CoffeeVMachine::run() { // 커피 자판기 작동  cout << "\*\*\*\*\* 커피자판기를 작동합니다. \*\*\*\*\*" << endl;  for (int i = 0; i < 3; i++) {  tong[i].fill(10);  }  int num;  bool check;  while (1) {  cout << "메뉴를 눌러주세요(1:에스프레소, 2:아메리카노, 3:설탕커피, 4:잔량보기, 5:채우기, 0:종료)>> ";  cin >> num;    check = checkInputError(num);  if (check == false)  cout << "입력한 메뉴는 존재하지 않습니다." << endl;    switch (num) {  case 1: // 에스프레소  selectEspresso();  break;  case 2: // 아메리카노  selectAmericano();  break;  case 3: // 설탕커피  selectSugarCoffee();  break;  case 4: // 잔량보기  show();  break;  case 5: // 채우기  fill();  show();  break;  case 0: // 종료  return;  }  }  }  int main() {  CoffeeVMachine cvm;  cvm.run();  return 0;  } |
| **[실행 결과]** |