1. Trace 방식 두 가지
   1. Step over
      1. 코드의 main 함수를 기준으로 tracing 하며, 코드 중 함수가 나오는 경우 함수 내 코드를 한 번에 tracing 한 뒤 main 함수의 코드를 따라가는 방식 즉 함수를 실행하되 내부로 들어가지 않음.
      2. Visual Studio에서 ‘F10’으로 진행 가능
   2. Step in
      1. 코드의 main 함수를 기준으로 trace 하다가 코드 중 함수가 나오는 경우, 함수 내 코드를 한 줄 한 줄 tracing 한 뒤 다시 main 함수의 코드로 돌아와 trace 하는 방식. 함수 내부로 들어 감.
      2. Visual Studio에서 ‘F11’키로 진행 가능
   3. Trace는 ‘Shift + F5’으로 종료 가능하다.
   4. 커서가 있는 위치부터 Trace를 진행하고 싶다면 ‘Ctrl + F10’으로 가능하다.
   5. Step Out: 현재 함수의 리턴으로 이동. 즉 현재 함수를 빠져나옴.
2. 배열 초기화 방법
   1. 배열 인덱스 하나하나 값 넣기
   2. 배열 선언 시점에 중괄호로 한 번에 초기화
   3. Int 배열의 경우 배열 선언 시점에 {0}로 초기화 하여 전체 배열 내 값을 0으로 초기화
3. 함수 개념
   1. 함수란 소스 코드를 묶어서 나만의 명령어를 만든 것.
   2. C++의 소스코드는 순차적으로 해석되기에 직접 만든 함수를 호출하기 위해서는 호출보다 선언이 선행되어야 한다. 즉, 함수 선언 코드가 함수 실행 코드보다 위에 적혀 있어야 한다.
4. 지역변수/전역변수 개념
   1. 함수 안에 만들어지는 일반 변수들을 지역변수라고 하며, 지역변수는 함수 내에서만 사용된다. 즉, 서로 다른 함수에서 같은 이름의 지역변수를 사용하더라도 서로 영향을 끼치지 않는다.
   2. 함수 밖에서 만든 변수를 전역변수라고 하며, 전역변수는 모든 함수들이 사용할 수 있다.
5. Char 변수 – 아스키코드
   1. Character 변수는 문자 자체를 저장하는 것처럼 보이나, 실제로는 문자에 해당하는 아스키코드 값을 저장한다. 따라서 for 반복문의 조건식에 char 변수를 넣어도 동작이 가능하다.
   2. 임의의 문자에 대한 아스키 코드를 알고 싶으면 문자 앞에 ‘(int)’를 기입하면 된다.
   3. (Tip) 대문자를 소문자로 바꾸는 방법(소문자는 반대로)  
      ex) char ch = ‘U’;  
       cout << (char)(ch-‘A’+’a’);
6. 구조체 쓰는 이유
   1. 구조체란 변수들을 묶어 새로운 Type을 만든 것이다.
   2. 구조체는 변수들을 묶은 형태로 그룹화할 때 사용한다.
7. Call by value / call by reference
   1. Call by value
      1. 함수 호출 시 전달한 매개변수의 값이 함수로 직접 전달되는 방식이다.
      2. 변수의 값이 복사되어 함수로 전달되기에 함수 내에서 변수 값을 바꾸더라도 원래 변수는 영향을 받지 않는다.
   2. Call by reference
      1. 함수 호출 시 매개변수의 주소값을 전달하는 방식
      2. 함수로 전달된 주소값과 원래 변수가 가진 주소값은 동일하므로, 함수 내에서 주소가 가리키는 값을 변경하면 원본 변수의 값도 변경된다.
      3. Call by reference로 변수 값을 변경할 때에는 결과값을 리턴할 필요가 없으며, 함수에 주소값을 던질 때에는 해당 값이 변하는 결과를 얻으리라 기대할 수 있다.
      4. 배열을 함수에 넣어서 보내면 함수는 이를 포인터로 받아들여 call by reference 개념으로 사용한다. 이 때 포인터가 가리키는 값은 배열 첫 번째 요소의 주소값이다.
8. 포인터 개념
   1. 포인터란 주소를 저장하는 변수이다. 변수 이름 앞에 ‘&’를 붙이면 변수의 주소값을 의미하는 것이다.
   2. 주소를 저장한다는 개념은 해당 주소를 가리키고 있다고 생각할 수 있다.
   3. 포인터 앞에 ‘\*’을 붙이면 가리킨 곳의 값을 나타내는 것이다.
   4. 포인터를 이용하면 함수에서 여러 값들을 결과값으로 받을 수 있다. (본래 return은 하나의 결과값만을 반환할 수 있다.)
9. 문자열(캐릭터 배열)
   1. 문자열은 여러 글자로 구성되어 있기에 char 배열로 저장 가능하다.
   2. Char 배열로 구성된 문자열은 문자열의 끝에 ‘\0’이라는 문자열의 끝을 나타내는 문자가 붙어야 한다. 만약 이 문자가 없을 경우 char 배열을 cout 하면 끝에 이상한 문자가 붙게 된다. 🡺 error
   3. Char 배열로 구성된 문자열은 아래와 같은 방법으로도 초기화할 수 있다. 이 경우 ‘\0’ 문자는 자동으로 붙게 된다(배열의 공간은 미리 만들어둬야 한다).
      1. Ex) char vect[4] = “abc”;
10. Cstring library
    1. ‘#include <cstring>’을 상단에 선언한 뒤 사용 가능한 라이브러리
    2. 세 가지 API
       1. Strlen(char배열): 배열 길이 구하기
       2. Strcpy(char배열1, char배열2): 복사(배열2🡺배열1)
       3. Strcmp(char배열1, char배열2): 비교(같은 경우 0을 리턴)
11. 선택정렬이란
    1. 배열의 요소들을 바로 옆의 배열 요소와 비교하여 기준에 따라 정렬하는 것.
12. 설계 및 구현 과정
    1. 과정 설계: 순차적으로 코딩해야 할 내용들을 계획(연습장, 주석 활용)
    2. 과정 하나씩 테스트: 설계한 내용을 구현할 때마다 Test 진행
    3. 과정별 주석 남기기: 코드 부분부분이 어떤 내용인지 빠르게 파악하기 위함
    4. 버그 찾아내기: Trace 이용
13. DAT
    1. 값을 인덱스로 활용하는 코딩 방식을 DAT(Direct Addressing Table)이라고 함
    2. 어떤 배열의 요소들이 어떤 문자들로 채워져 있는지 확인하거나 문자 개수를 셀 때 활용할 수 있다.
14. Direct 코딩 기법
    1. Direct라는 offset 배열을 이용하여 배열의 일부분을 효율적으로 순회할 수 있다.
    2. 예를 들어 2차원 배열 A가 있고 A에 있는 임의의 요소 주변 값들을 합하라는 문제가 있을 경우, direct라는 배열을 만들어 요소의 주변에 해당하는 y와 x값 쌍을 요소로 넣어두고, 반복문을 통해 A의 전체 범위를 순회하는 것이 가능하다.
    3. <레벨 19의 isPattern 함수 참고>
15. 재귀함수 사용 방식
    1. 재귀함수란 함수 안에서 자기 자신을 호출하는 함수를 말한다.
    2. 잘못 사용하면 무한 재귀현상을 야기할 수 있으므로 재귀함수에 해당하는 함수 내에 함수를 조건부 중간 종료시킬 수 있는 코드가 포함되어야 한다.
16. 재귀호출 path
    1. 재귀호출 할 때 어떤 경로로 들어와 있는지 확인하는 방법으로 사용된다.
    2. 전역배열에 이력을 기록하여 경로를 확인하거나 Trace하는 것이 가능하다.
    3. 함수호출을 하기 직전에 Path에 다음 행동을 기록하고, 함수 호출이 끝나면 기록을 지워주는 방식으로 진행한다.
17. 알고리즘 성능

A.





