두 가지 방법으로 stack 구현

5846593 이지현

구조체 배열로 구현한 스택은 스택에 저장되어야 하는 값이 정수나 문자가 아니고 더 복잡한 구조를 갖는 요소일 때 사용이 됩니다. 구조체 안에 필요한 모든 정보를 넣으면 되기 때문입니다. 구조체 배열로 구현 할 시 top과 stack배열을 하나의 구조체를 결합시키고 이 구조체의 포인터를 함수로 전달합니다. 즉 StackType이라는 새로운 구조체 타입을 만들고 여기에 stack배열과 top을 넣습니다. 그리고 이 구조체에 대한 포인터를 각 함수의 매개변수로 전달하는 것입니다. 모든 함수에서 만약 전달된 포인터가 s이면 기존의 top이라고 사용하던 것을 s->top으로 변경하면 됩니다. stack배열도 같은 방법으로 사용하면 됩니다. 이렇게 되면 전역변수로 구현하는 것과 다르게 쉽게 여러 개의 스택을 만드는 것이 가능해집니다. 이러한 방법은 포인터를 이용하여 스택을 조작합니다. 이것은 구조체를 함수의 매개변수로 전달하였을 경우, 구조체의 원본이 전달되는 것이 아니라 구조체의 복사본이 함수에 전달됩니다. 따라서 원본을 수정할 시에는 포인터를 사용해 변경할 수 있습니다. 구조체 배열을 이용하면 스택의 데이터를 구조체의 멤버 면수로 저장할 수 있어 데이터를 쉽게 조작할 수 있지만 스택의 크기가 미리 정해져 있기 때문에, 스택이 가득 차면 더 이상 데이터를 저장할 수 없게 됩니다. 스택을 동적 메모리 할당으로 생성하는 방법을 사용할 시 이를 보완할 수 있습니다. 이 방법을 사용하면 각종 함수들을 호출할 때 보다 자연스러운 표현이 가능합니다. 하지만 사용이 끝나면 반드시 동적 메모리를 반환해야 합니다. 동적 메모리 할당을 사용할 시 필요할 때마다 스택의 크기를 동적으로 늘릴 수 있습니다. 메모리를 늘리는 방법은 push()에 realloc()함수를 사용하여 현재 내용을 유지하면서 주어진 크기로 동적 메모리를 다시 할당할 수 있습니다. 이 방법은 스택의 크기가 동적으로 변하는 상황에서는 유리하지만, 메모리 관리 등의 부담이 있습니다.