*#include* <iostream>

int main(){

int n =3;

int\* pn =&n;

char c = 'A';

char\* pc = &c;

*//value*

printf("%d %p\n", n, pn);

*//address*

printf("%p %p\n", &n, &pn);

printf("%d %d\n", sizeof(n), sizeof(pn));

*//value*

printf("%c %p\n", c, pc);

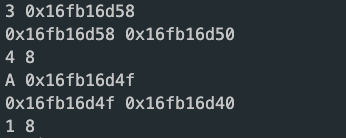
*//address*

printf("%p %p\n", &c, &pc);

printf("%d %d\n", sizeof(c), sizeof(pc));

*return* 0;

}



1. 위의 코드에서 왜 pc의 크기는 각각 8byte 인가?

답 : c언어의 경우 포인터 변수의 크기는 4byte 라고 하지만, 64bit 컴퓨터의 경우 메모리의 주소공간이 2^64인 8byte가 된다.

2. 어떤 문자열이 팰린드롬인지 판별하는 방법은 무엇입니까?

펠린드롬이란 ‘eye’같이 거꾸로 읽어도 똑같은 단어나 문장을 말한다

입력된 문자열에서 앞의 문자와 뒤의 문자가 같은지 하나씩 비교해보았을때, 모든 글자가 같으면 그 문자열은 펠린드롬이다.

3. "스택(Stack)이 왜 일반적으로 컴파일러나 인터프리터에서 사용될까요?"

스택의 특징이 후입선출(Last-In-First-Out, LIFO)의 자료구조로, 새로운 데이터가 항상 최상단에 쌓이고, 가장 최근에 추가된 데이터부터 차례로 꺼내어져서 사용된다. 이러한 특징 때문에 컴파일러나 인터프리터에 이용됩니다.

스택의 쓰임새(컴파일러와 인터프리터)

함수 호출과 반환

컴파일러나 인터프리터에서는 함수 호출 및 반환을 스택을 이용하여 관리합니다. 함수 호출 시 매개변수와 지역 변수를 스택에 push하고, 함수 반환 시 스택에서 pop합니다. 이를 통해 함수의 실행 흐름을 관리하고, 지역 변수의 범위를 제한할 수 있습니다.

수식 계산

컴파일러나 인터프리터에서는 수식 계산을 위해 스택을 사용합니다. 중위 표기법(infix notation)의 수식을 후위 표기법(postfix notation)으로 변환하고, 스택을 이용하여 후위 표기법의 수식을 계산합니다.

블록 구조

컴파일러나 인터프리터에서는 블록 구조(block structure)를 구현하기 위해 스택을 사용합니다. 블록 구조는 코드의 가독성과 유지 보수성을 높이는 데 중요한 역할을 합니다. 스택을 이용하여 블록의 시작과 끝을 추적할 수 있습니다.

메모리 할당

컴파일러나 인터프리터에서는 메모리를 할당하고 해제하는 데 스택을 사용합니다. 동적 할당된 메모리는 스택에 push되고, 해제되는 순서는 스택에서 pop되는 순서와 반대로 이루어집니다.

따라서, 스택은 컴파일러나 인터프리터에서 함수 호출, 수식 계산, 블록 구조, 메모리 할당 등 다양한 기능을 구현하는 데 매우 유용한 자료구조입니다.

스택은 구조가 단순해서 구현이쉽고, 데이터 저장/읽기 속도가 빠르기때문이다.

컴파일러란?

전체 소스 코드를 한 번에 읽어와 기계어로 번역한 후, 그 결과물을 실행하는 방식입니다. 즉, 소스 코드를 전처리하고, 분석하고, 번역하는 과정을 거친 뒤에 실행 가능한 바이너리 코드를 만들어냅니다. 이 바이너리 코드는 실행 속도가 빠르지만, 번역된 코드를 수정하려면 다시 소스 코드를 수정하고 컴파일을 해야 하기 때문에 수정이 번거로울 수 있습니다.

인터프리터란?

소스 코드를 한 줄씩 읽어와 바로바로 실행하는 방식입니다. 즉, 소스 코드를 한 줄씩 읽고 해석한 후, 실행합니다. 이 방식은 실행 속도가 느리지만, 코드를 수정하는 경우에는 소스 코드를 수정하고 바로 실행할 수 있기 때문에 수정이 용이합니다.

처리방식에 따른 다른 점

언어의 처리 방식에 따라서 컴파일러와 인터프리터가 선택되기도 하고, 또한 언어에 따라서 컴파일러와 인터프리터 중 하나를 선택할 수도 있습니다. 일반적으로 C, C++ 등과 같은 언어는 컴파일러를 사용하고, Python, Ruby 등과 같은 스크립트 언어는 인터프리터를 사용합니다.

Java는 일반적으로 컴파일러와 인터프리터를 함께 사용합니다.Java는 소스 코드를 컴파일하여 바이트 코드로 변환하는 과정에서 컴파일러를 사용하고, 이후 바이트 코드를 실행하는 과정에서는 인터프리터를 사용합니다. 이를 통해 Java는 높은 이식성, 안정성 및 보안성을 제공할 수 있습니다.

4. Delete와 new의 역할은?

new 연산자는 특정 데이터에 대한 메모리 공간을 동적으로 할당합니다.

delete 연산자는 동적으로 할당된 메모리 공간을 해제합니다.

new 연산자를 사용했으면 이후에 delete 연산자를 사용해야한다. 그렇지 않으면 메모리 누수가 발생한다.

new: 메모리 공간 동적으로 할당(제한두지않음)-new로 할당받은 메모리는 크기 변경 불가능(변경하고 싶을 경우원하는 크기로 새로 할당, 복사. 이전 메모리는 해제)

delete: 동적 할당 해제(초기화)

C에는 없었으나, C++에서 동적 메모리 할당을 위하여 추가된 연산자이다. (C에서는 malloc함수로 구현하였으나 C++의 delete, new연산자의 표기가 더 간결하여 C++에서는 malloc을 잘 사용하지 않는다.)

new는 힙에 메모리 할당을 해주고 생성자를 호출해주며 해당 타입으로 변환을 해준다.

delete는 소멸자를 호출하고 메모리를 해제해준다.

자료구조 2주차 질문하고 싶은 문제

19, 21, 22, 25,