

Function

2022년 5월 18일 수요일 오전 11:43

Subroutine

1. Value returning function
 - a. Function
2. Non value returning function
 - a. Void function
 - b. Procedure == void

Parameter : formal parameter, 함수 선언에서 나타나는 표현

Argument : Actual parameter, function call에서 나타나는 표현
Function caller에서 호출하는 함수에 parameter를 argument라고 하는듯!

```
Void f(int a) //parameter
{
}
Int main()
{
    f(5); //Argument
}
```

Pass by

- **Value**
 - Argument를 미리 계산하고 값을 parameter로 넘김
 - 그래서 callee argument를 바꾸어도 parameter에 영향을 주지 않는다
 - 단 배열 같은 경우는 주소값을 넘긴다
- **Reference**
 - 반드시 L- Value여야한다
 - Caller는 argument의 주소를 넘긴다
 - Callee가 caller의 환경을 바꿀수가 있다 [장점이자 단점]
- **Value result**
 - Call by value로 넘기고 계산 다 끝내고 result를 넘긴다
 - 멀티 스레드 시스템에서 실행 되는 경우
- **Result**

Pass by Value-Result and Result

- Pass by value at the time of the call and copy the result back to the argument at the end of the call.
 - *E.g., Ada's in out parameter*
 - *Value-result is often called copy-in-copy-out.*
- Reference and value-result are the same unless the value of an argument can be examined before returning to the caller.

화면 캡처: 2022-05-18 오후 12:36

- Name
 - Late binding
 - Lazy evaluation
 - $x+y$ 라는 parameter 있으면 텍스트 자체로 넘기고 계산을 동적으로 처리한다
 - 동적으로 parameter 계산한다

Language Examples

1. **FORTRAN**
pass by value-result, and by reference
2. **ALGOL 60**
 - Pass-by-name is default;
 - pass-by-value is optional
3. **Pascal**
 - pass-by-value is default;
 - pass-by-reference is optional.
4. **C**: Pass-by-value except array parameter
5. **C++** : Like C, but also offers reference type parameters (&)
6. **Java** : Like C, but references are used for variables for objects

화면 캡처: 2022-05-18 오후 12:40

연습 문제 예시

Parameter Passing Examples

```
program test {  
  l: integer;  
  procedure R(x:integer) {  
    x:=x+1;  
    write(x);  
  }  
  
  l:=5;  
  R(l);  
  write(l);  
}
```

실행 결과

by value:	6, 5
by reference:	6, 6
by name:	6, 6
by value-result:	6, 6

화면 캡처: 2022-05-18 오후 12:41

오른쪽 : 함수값을때의 값

왼쪽 : l 값

Caller / Callee

Function call/return

function의 call/return 의미

Caller

- allocate storage for locals
- pass parameters
- save the return address
- transfer control to the callee

Callee

- compute the result using parameters, locals, and nonlocals
- save the result
- return to the caller

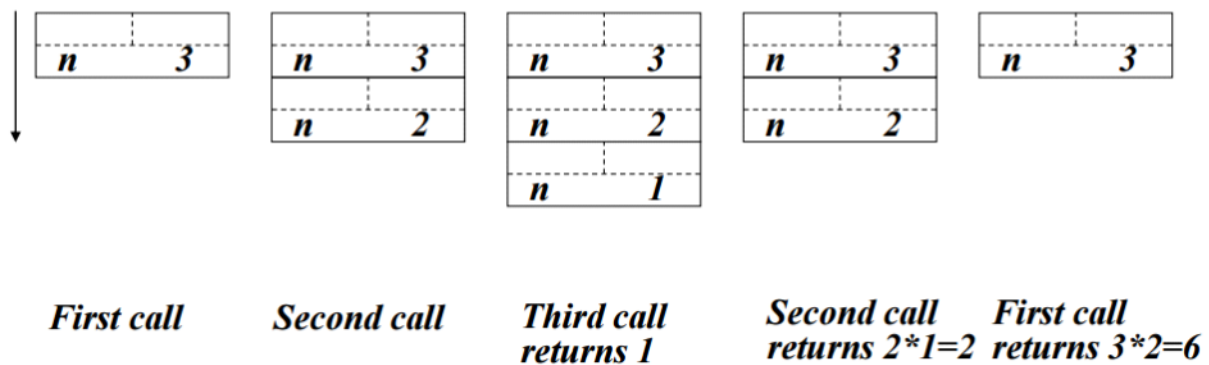
화면 캡처: 2022-05-18 오후 12:45

구현

- 함수 구현 하려면 실행 시간에 필요한 메모리 공간(**Activation Record** - AR)을 할당해야한다
- AR에는 함수 실행에 필요한 parameters, local variable, result등의 정보를 저장한다
- AR은 함수가 호출시 할당되고 종료시 삭제
- AR을 할당하기 위해 설정해 놓은 메모리 영역을 **Run time Stack**이라고 한다
 - AR은 runtime stack에 할당 된다

Recursion function

Run-time Stack Behavior for factorial(3)



화면 캡처: 2022-05-18 오후 12:52

Runtime Memory Organization

실행시간 메모리 내용

Program Code

- user code
 - executable machine code
- runtime support
 - library routines - i/o functions, sin, cos, ..
 - storage management:
 - allocation/deallocation, garbage collector
 - interpreter code (Lisp, JVM, ...)

Data

- variables : static, local, dynamic variables
- environment information, parameters,
- function call/return, function result

화면 캡처: 2022-05-18 오후 3:55

Memory Allocation

1. Static Allocation

- a. 실행전에 메모리에 할당한다
- b. Program code
- c. Static variables

2. Dynamic allocation

- a. 실행시간에 필요시 할당
- b. Run time stack
- c. Activation record 할당 /반환
 - i. 미리 알수가 없기 때문에 dynamic으로 처리
- d. Heap storage(동적 메모리)
 - i. C : malloc()/free()
 - ii. C++ : new / delete
 - iii. Java : 자동관리
 - 1) 객체를 heap에 생성
 - 2) 객체 삭제는 automaic garbage collection 사용

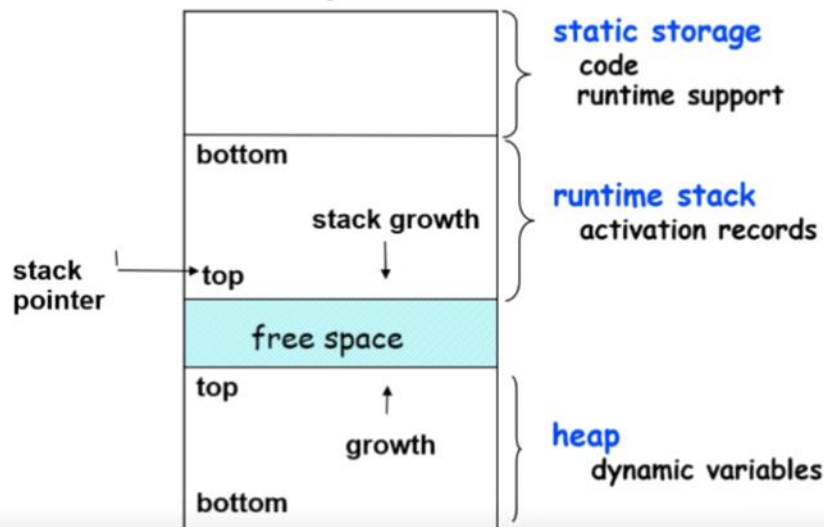
Pascal은 함수안에 함수가 들어갈수있다 하지만 c는 그렇게 안됨
그래서 메모리 관리가 c보다 까다로움

Pascal

- Parameters are passed by value,reference
- Local variavles, Recursion ,nested functions [지역 변수, 재귀함수 중첩함수] 지원
- Static scoping
 - X 변수의 binding이 dynamic scoping에서는 호출하는 함수에서 x의 바인딩을 찾는다
- 동적 메모리 할당 해제 가능

실행시간 메모리 구조 - Pascal

Pascal의 Memory 구조

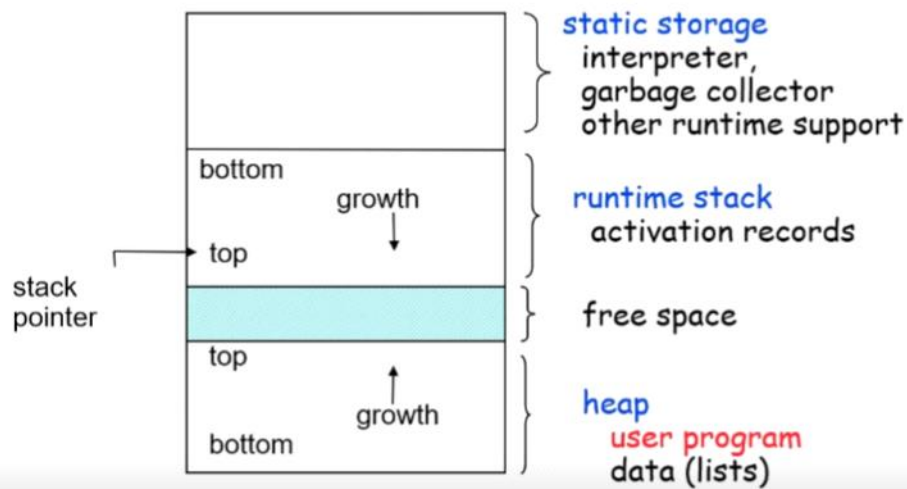


화면 캡처: 2022-05-21 오후 7:26

Static은 어느정도 할지 아니까 미리 저장 공간 결정 할수있고 runtime stack,heap은 모르니까
Run time stack은 아래로 가고 heap은 위로 간다

LSIP 메모리 구조

실행시간 메모리 구조 - LISP



화면 캡처: 2022-05-21 오후 7:59

프로그램이 list로 이루어져 있다

Functional program

C에서 global , local , static variable의 차이점

Global 변수는 외부에서 참조 할수있다 extern명령어 사용하면 됨

Static 변수는 global과 차이가 무엇인가? : 외부에서 볼수없는 변수이다. 파일 내부에서만 글로벌처럼 사용할수있다

Parameter는 함수의 지역 변수와 같다

함수안에 static은 지역변수와 다르다.

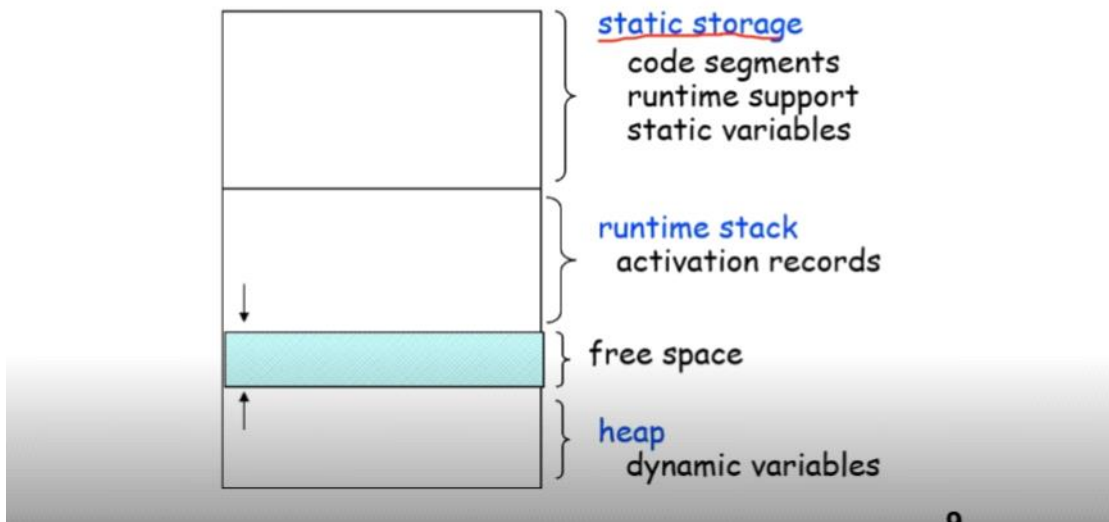
지역변수는 {} namespace 끝나면 life time 끝나지만 static은 영구적이다 다른 메모리에 따로 저장 되어있다

Static variable은 static storage에 저장이 된다.

Local 변수는 AR에 저장이 된다 -> runtime stack

실행시간 메모리 구조 - C, C++

C, C++: No nested subprograms



화면 캡처: 2022-05-21 오후 8:17

Static storage는 compiler가 미리 알수있다 -> 미리 할당 할수있다. Static변수나 global변수들은 static storage에 저장된다

Activation Record -> runtime stack에 저장 된다

Subprogram call , 실행, return을 위한 자료 저장소

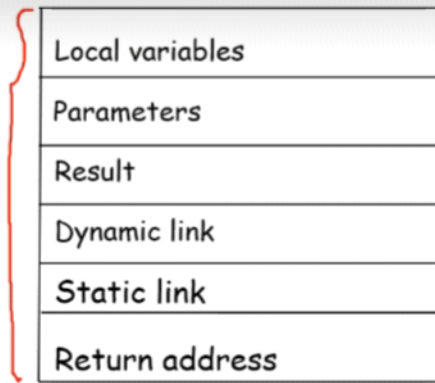
구조는 함수마다 다르다

-> 그러나 static link, dynamic link, return address는 동일하게 들어갈 것이다.

AR구조

- Local variables (locals)
- Parameters
- Return address
- Result
- Environment information(static link, dynamic link)

Activation Records



Local variables
Parameters
Result
Dynamic link
Static link
Return address

- subprogram이 호출될 때 runtime stack 에 activation record 생성
- dynamic link: 동적 부모(the caller) 의 activation record를 가리키는 link. return시 caller의 실행환경 복구
- static link: 정적 부모(static parent)의 activation record의 link (nonlocals 접근 구현)

화면 캡처: 2022-05-21 오후 8:27

Subprogram이 호출될때 runtime stack에 activation record 생성 해야한다

Static link : F함수 안에 G함수가 있다면 G함수의 activation record는 F함수의 AR을 가리키도록 하는거
정적 부모(static parent)의 activation record의 link(nonlocals 접근 구현)
다른 AR을 가리키는 link

Dynaminc link : 동적 부모(the caller)의 activation record를 가리키는 link. Return시 caller의 실행 환경 복구
자신을 호출한 함수의 AR가리키는 link?

위의 3개는 함수의 정의에 따라 달라짐 아래이 3개는 어느 함수든지 같다

Call - Statement
Value return function -> Expression

F1->f2->f3->f4를 호출한다고 하면
F2의 dynamic link는 f1을 가리키고 있고
F3의 dynamic link는 f2를 가리키고 있다
이런식으로 진행

만약 f2가 f1내부에 정의된 함수라면 f2의 static link도 f1을 가리키고 있다

F3,f4 모두 f1내부에 정의 된 함수라면 f2처럼 f3,f4모두 f1을 static link로 가진다

구현

Nolocal 참조의 구현

Issue: How to bind a reference to a nonlocal variable?

⇒ 참조된 변수가 어느 Activation Record에 존재하는가?

static link: static parent의 AR에 접근을 위해 필요한 주소값.

정의:

static chain: static link들로 이루어지는 chain

화면 캡처: 2022-05-21 오후 8:55

-> 중첩 함수를 구현하기 위해 필요한 기술 nlocal

```
f()
{
    g()
}
```

g()의 static link는 f()를 가리킬 것이다. 이런식으로 static link들로 이루어지는 static chain으로 AR 참조 해야한다

Example Program

```
program MAIN;
  var X : integer;
  procedure BIGSUB;
    var A, B, C : integer;

    procedure SUB1;
      var A, D : integer;
      begin A := B + C; end;

    procedure SUB2(X : integer);
      var B, E : integer;
      procedure SUB3;
        var C, E : integer;
        begin { SUB3 }
          SUB1;
          E := B + A;
        end; { SUB3 }

      begin { SUB2 }
        SUB3;
        A := D + E;
      end; { SUB2 }
    begin SUB2(7); end; { BIGSUB }

  begin BIGSUB; end. { MAIN_2 }
```

```
MAIN
=> BIGSUB
=> SUB2
=> SUB3
=> SUB1
```

화면 캡처: 2022-05-21 오후 9:01

Main은 BIGSUB을 호출 가능 근데 sub1은 bigsub의 지역함수이기때문에 호출 불가
Sub1은 bigsub도 보이고 main 함수도 보인다

A의 변수는 bigsub에도 있고 sub1에도 있는데 sub2실행 중에 A가 어딘는지 알아내는게 어려움이 있다

Sub3의 A는 bigsub의 A이고 sub2의 A는 Bigsub의 A이다 -> 어떻게 바인딩 하는가? -> static link로 binding 한다 어떻게?
아래에 설명이 있다

Nesting depth

e.g.

```
graph LR
    main[main] --- A[A]
    A --- B[B]
    main --- C[C]
```

main ----- depth = 0
A ----- depth = 1
B ----- depth = 2
C ----- depth = 1

nonlocal 참조의 구현

chain_offset = 참조가 일어난 함수의 깊이 - 변수가 정의된 함수의 깊이

local_offset = AR 내에서의 변수의 상대 위치

컴파일 시간: **nonlocal variable**의 참조를 (chain_offset, local_offset)로 **binding**

실행 시간: 참조된 **nonlocal** 변수는 **static chain**을 따라 **chain_offset** 만큼 떨어진 **AR**에 존재한다.

화면 캡처: 2022-05-21 오후 9:12

바깥의 함수 깊이가 0이라 하자

Chain offset :

예) 참조가 일어난 함수의 깊이 - 변수가 정의된 함수의 깊이
 $2(B) - 0(\text{main})$

Local offset : AR내에서의 변수의 상대 위치

컴파일 시간

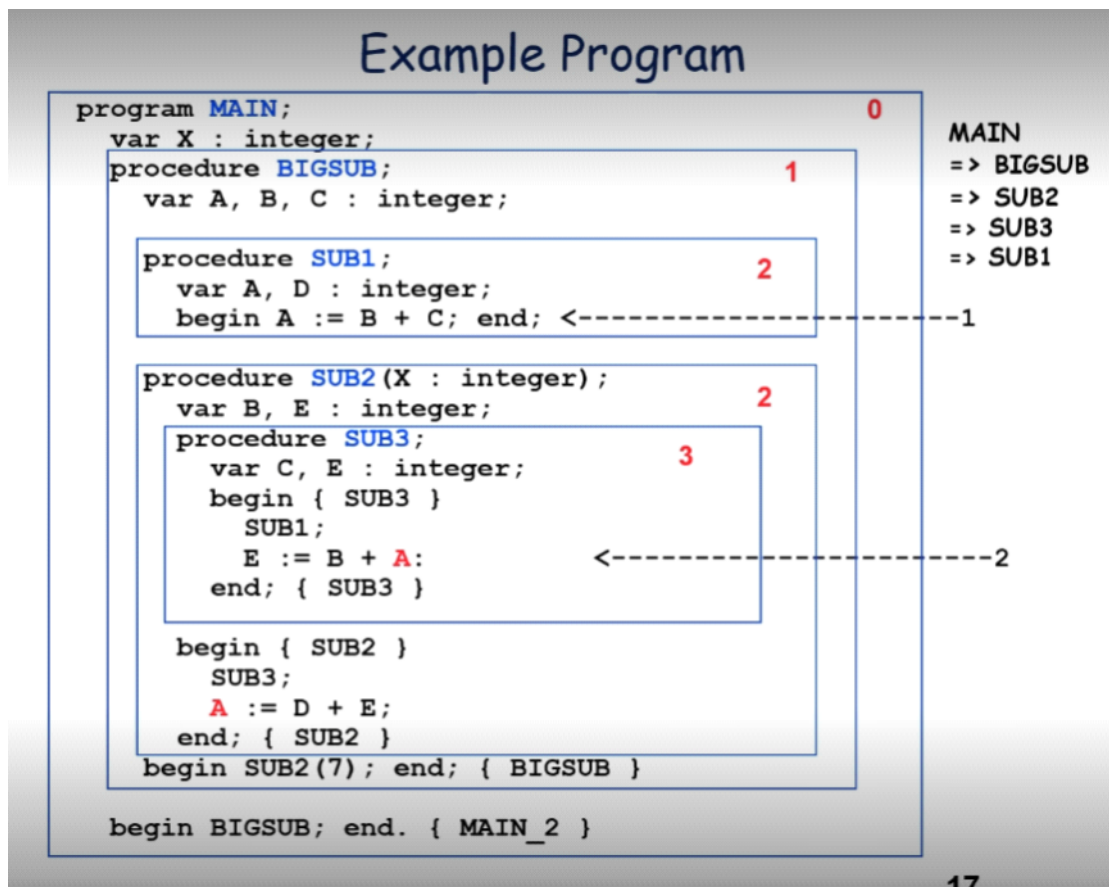
Nonlocal variable의 참조를 (chain offset, local offset)로 binding

실행 시간

참조된 nonlocal 변수는 static chain을 따라 chain offset 만큼 떨어진 AR에 존재한다

실행중일때 AR를 어떻게 찾아갈까 -> chain offset만큼 static link 타고 들어가서 local offset만큼 떨어져 있는곳에 있다

Copile시간에 (chain offset, local offset)이 계산 됨 그거 보고 들어가면 됨



화면 캡처: 2022-05-21 오후 9:18

Static Scoping 구현

Call sequence for MAIN
MAIN =>BIGSUB =>SUB2=>
=> SUB3 =>SUB1

At position 1 in SUB1:

$$A = (0, 3)$$

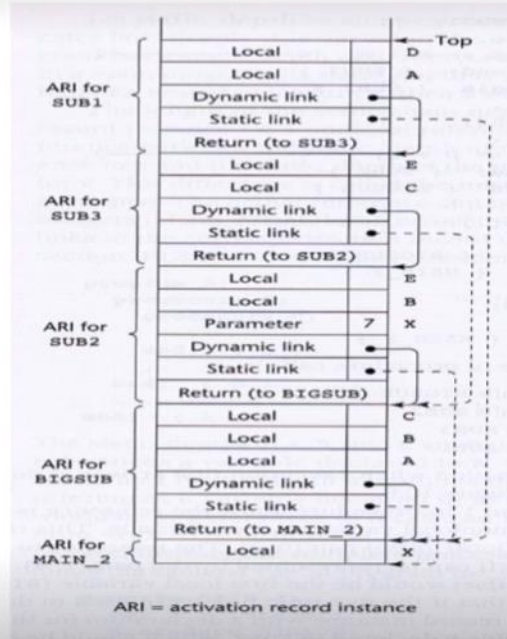
B - (1, 4)

 $C = (1, 5)$

At position 2 in SUB3:

 $E = (0, 4)$
$$B = (1, 4)$$

$A = (2, 3)$



화면 캡처: 2022-05-21 오후 9:19

함수 sub1이 실행 되고 있을때 run time에 (chain offset, local offset)를 통해 각 변수가 어느 위치에 있는지 알수있다

구현한 프로그램에서 중첩 함수는 구현이 안되어있다

그리고 실제 구현한 프로그램에서는 static link를 따라가서 변수를 찾는 구조이다 없으면 또 연결되어있는 static link가서 탐색하는 구조