**Design and Development of Compiler**

**for C- Language**

**Phase 4: A Code Generator**

**1. project4 결과 보고서**

**과목명: [CSE4120] 기초 컴파일러 구성**

**담당교수: 서강대학교 컴퓨터공학과 정 성 원**

**개발자: 7조**

**고원섭(팀장,111584)**

**정다은(131602)**

**이서희(141555)**

**개발기간: 2017. 06. 20 - 2017. 06. 28**

**Project 4 결과 보 고 서**

**프로젝트 제목: Design and Development of Compiler for C- Language:**

**Phase 4: A Code Generator**

**제출일: 2017. 06. 28.**

**참여조원: 고원섭(팀장), 정다은, 이서희**

**I. 개발 목표**

**-이전 프로젝트에서 구현한 C- Semantic Analyzer를 활용하여 SPIM machine에서 동작하는 code를 생성하는 C- 용 code generator를 개발한다.**

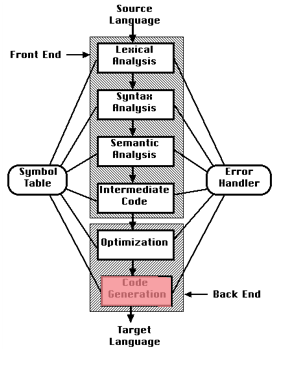
**-이는 메모리를 stack 기반으로 access하며 Parse Tree의 한 node 마다의 kind에 따른 instruction을 작성한다.**

**-global & local 변수의 stack에 쌓는 방식을 다르게 접근해야 한다.**

**-해당 Scope의 종료 시에 stack의 memory를 지워줘야 한다.**

**II. 개발 범위 및 내용**

**가. 개발 범위**



-appendix B에 있는 code.h, code.c, cgen.h, cgen.c를 C-에 맞게 수정한다.

i. SPIM machine에서 구동 가능한 [test].s SPIM code를 생성한다.

ii. i에서 code 생성에 사용할 Utility를 작성한다.

**나. 개발 내용**

i. 생성된 코드를 SPIM에서 console 창을 이용하여 input / output을

수행하기 위해system call을 명시한다.

ii. generateCode( )함수를 생성하여, node마다 분석한 결과를 이용하여

각 kind에 맞는 함수를 호출한다.

iii. 각 kind에 맞는 함수에서 각 역할 별로 SPIM machine에서 generate되는

instruction을 print한다.

**III. 추진 일정 및 개발 방법**

**가. 추진 일정**

|  |  |
| --- | --- |
| 기간 | 내용 |
| 17.06.19~21 | 역할 분담 및 설계 |
| 17.06.22~26 | 프로그램 제작 |
| 17.06.27~28 | 보고서 작성 |

**나. 개발 방법**

* 이전 프로젝트에서 제작한 Semantic Analyzer를 바탕으로 code generator for c- 를 작성했다.
* semantic error check를 모두 통과한 경우 메인 함수에서 codeGen 함수를 호출하여 code generation이 시작된다.
* 사용한 함수들은 cgen.c와 code.c에 작성했다. cgen.c는 실질적으로 코드를 생성하는 부분이고 code.c에는 fprintf 부분을 래핑한 utility 함수이다.
* .s 파일을 생성하여 spim으로 테스트했다.
* 교과서에 배포되는 appendix의 코드는 Tiny 언어에 맞게 작성되어 있으므로 C- 언어에 적합하게 기존 함수 및 자료구조를 변경해야 한다.
* Linux 환경의 cspro 9 서버에서 프로그램을 작성한다.
* 각 조원들이 구현해야할 파트를 나누어서 코딩하고 개발 내용을 연동 및 통합한다.
* 구글 드라이브 및 메신저를 이용하여 진행사항을 보고하고 확인한다.   
  [Compiler]project\_4\_Appendix\_7
* 테스트를 통해 각종 예외사항이나 오류상황들을 확인하고 해결방안을 찾는다.

**다. 연구원 역할 분담**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **구분** | **내용** | **기여도(%)** | | |
| **고원섭** | **정다은** | **이서희** |
| **설계** | 기초 배경지식 이해 | 33.3% | 33.3% | 33.3% |
| 일정 산출 | 33.3% | 33.3% | 33.3% |
| 개발환경 구축 | 33.3% | 33.3% | 33.3% |
| **개발** | input/output & Call & Return & Compound & function & array Declaration & var declaration & const & parameter passing | 33.3% | 33.3% | 33.3% |
| Variable & Subscription & Relop & If & While & Emit instruction & Emit push/ pop | 20% | 20% | 60% |
| Assign & Local Variable Declaration & Arithmetic & Comment print & instType function & Function Call & emit Function | 60% | 20% | 20% |
| **테스트** | 단위 테스트 (오류 체크) | 33.3% | 33.3% | 33.3% |
| 통합 테스트 | 33.3% | 33.3% | 33.3% |
| **산출물 작성** | proposal | - | - | - |
| result | 33.3% | 33.3% | 33.3% |
| **종합** | 전체 기여도 | 35.0% | 30.0% | 35.0% |

**IV. 연구 결과**

1. **합성 내용**:

**[각 부분의 역할 및 사용 기술]**

Project3의 Semantic Analyzer로부터 분석된 결과를 토대로

code를 generate한다.

이 프로젝트에서 사용한 함수들은 cgen.c와 code.c에 주로 있다.  
 cgen.c에 있는 함수들은 실제 코드를 생성하는 부분이고

code.c에 있는 함수들은 cgen.c에서 코드를 생성하는 데에 도움을 주는

Utility 함수들이다.

**[툴]**

Project3와 같이 Flex와 Bison을 사용하였고 QtSpim을 이용하여

.s 파일을 run했다.

**[사용자 매뉴얼]**

make로 컴파일을 한 뒤 다음과 같이 command를 입력한다.

./project4\_7 [테스트 파일명].c

command로 프로그램이 실행되면 code generation을 시작한다는 메시지와 함께

생성될 코드 파일 명을 표시해 준다. 코드를 생성하는 동안 standard output과

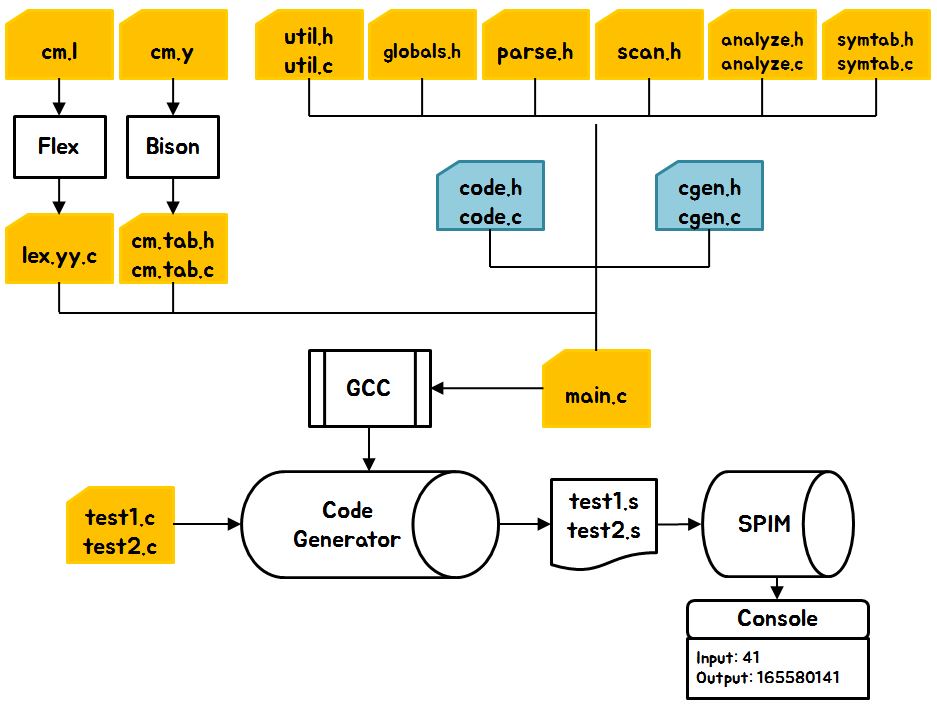
코드 파일에 동시에 코드를 출력해준다. 출력이 끝나면 code generation이

완료되었다는 메시지를 출력하고 끝난다.

SPIM 머신 코드가 출력된 [테스트 파일명].s 파일을 QtSpim machine에서 불러와

Run 하면 Console에 [테스트 파일명].c 코드의 input명령 또는 output이 출력된다.

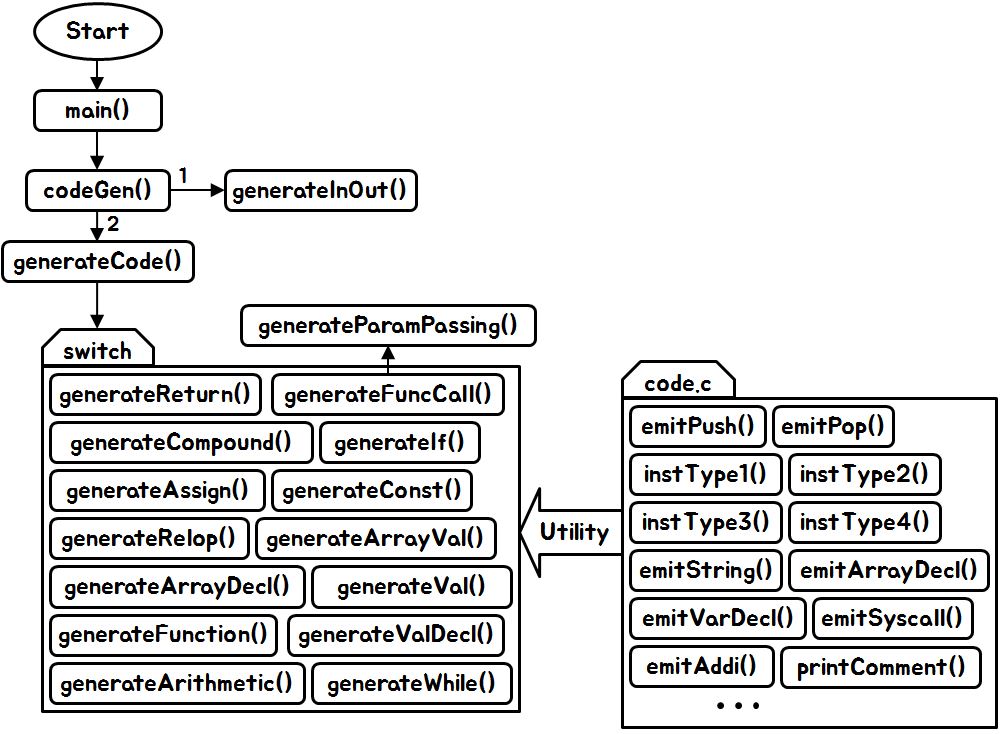
**[전체 소프트웨어 구성도]**



2. **분석 내용**:

**[알고리즘 구성 요소]**

|  |  |
| --- | --- |
| cgen.h  cgen.c | 실제 코드를 생성하는 부분이다.  c- 언어에서 static scope rule을 실현하기 위해 꼭 필요한 frame pointer를 포함하도록 구현했다. |
| code.h  code.c | cgen.c에서 코드를 생성하는 데에 도움을 주는 유틸리티 함수들 |



[프로그램 흐름도]

* generateInOut함수는 input함수와 output함수를 생성하기 위해   
  data section에 두 함수에서 사용할 string을 선언한다.   
  이 string으로 입력을 받을 때에 출력할 string인 \_prompt0와   
  결과값을 출력할 때 사용할 \_output0, newline인 \_linefeed0가 있다.  
  symbol과 이름이 겹치는 일을 방지하기 위해 이름 뒤에 0을 붙였다.  
  c-언어의 symbol들은 이름에 숫자가 포함될 수 없기 때문이다.  
  input과 output 함수는 symbol table의 전역 scope에 추가되어 semantic analyzer에서 정상적으로 동작하도록 했다.
* generatecode함수는 AST node를 입력받아 node type에 따라 분류하여  
  generateReturn(), generateFuncCall(), generateCompound(), generateIf(),  
  generateAssign(), generateConst(), generateRelop(), generateArrayVal(),  
  generateArrayDecl(), generateVal(), generateFunction(), generateValDecl()  
  generateArithmetic(), generateWhile() 중 하나를 호출한다.
* generateArrayDecl과 generateVarDecl함수는 전역변수 선언에 대해 호출되어 선언을 만든다.
* array의 경우 “\_name .word 0:size” 형태이고, val은 “\_name .word 0”형태이다.  
  이 때, 이름 앞에 ‘\_’를 붙이는 이유는 mul이나 div같은 instruction과 이름이 겹치게 되면   
  label이 아닌 instruction으로 인식하여 에러가 발생하기 때문이다.   
  위의 string들도 ‘\_’를 붙이는 이유는 이것과 동일하다. 0으로 인해 instruction과 이름이 겹치지 않지만 동일한 routine으로 처리하기 위해 ‘\_’가 붙도록 했다.  
  함수도 마찬가지로 앞에 ‘\_’가 붙는데 main함수는 entry point이므로 외부에서 호출할 수 있도록 하기 위해 예외처리 하여 ‘\_’가 붙지 않도록 했다.
* generateVal은 ID를 만났을 때 address를 가져와 left value일 시 v0에, 아닌 경우 address로부터 값을 가져와 v0에 담는다.   
  전역일 경우 la를, 지역일 경우 fp+offset을 이용하여 구했다.   
  만약 parameter로 array가 전달된 경우는 그 위치에 저장된 값을 주소로 사용해야 하므로   
  lw를 이용하여 값을 얻어온 뒤 그것을 주소로 사용하였다.
* generateConst는 상수를 v0에 넣는다. generateArithmetic과 generateRelop는 각각 산술과 관계 연산자에 대한 코드를 만든다.   
  먼저 left(child[0])와 right(child[1])에 대해 generateCode를 호출하여 값을 얻어 온 후   
  두 레지스터를 이용하여 값을 구한다. left를 구한 후에 right를 구할 때 레지스터 값이 변할 수 있으므로 left에서 구한 값을 스택에 넣어둔 후 right를 구한 다음 pop을 했다.
* generateArrayVal은 array subscription code를 만드는데 ID에 해당하는 array 시작 주소와 index를 각각 재귀 호출을 통해 구한 뒤에 array 시작주소 + INT\_SIZE(4)\*index를 통해 주소를 구한다. left value의 경우 그 주소를 그대로 반환하고 (v0), 아니라면 lw를 이용하여 그 주소에서 값을 가져온다.
* generateIf와 generateWhile은 각각 if문과 while문을 생성한다.   
  - if문은 child[0]에 대해 generateCode를 호출하여 condition 부분을 만든다. condition의 결과가 0이나 1로 v0에 저장된다.   
  그 값이 0일 시에 else가 있다면 else 부분으로 점프하고 else가 없으면 END 부분으로 점프한다. 그 값이 1이라면 그 다음 부분인 then 부분을 실행하게 된다.   
  그러면 generateCode를 child[1]에 대해 호출하여 then 부분의 코드를 만든다.   
  else가 있다면 then이 끝난 후 else가 실행되지 않도록 end로 점프하게 한 후 generateCode를 호출해서 else 부분을 생성한다.   
  - while문은 child[0]를 호출하여 condition 부분을 만든다.  
  condition의 결과가 0이면 end 부분으로 점프하게 하고, generateCode로 while body 부분을 생성한다.   
  이 부분이 끝나면 while문의 시작부분으로 점프하게 해서 다시 condition을 확인하도록 한다. 점프할 Label들은 name + number 형태로 이름이 부여되는데, 이 number가 각 if문이나 while문 마다 고유한 번호를 사용하게 해서 label이 겹치지 않도록 하였다.
* generateFunction은 함수를 만드는데, 함수 이름 label을 만들고 나서 다음과 같은 작업을 수행한다.

1. fp를 push하고, fp를 현재 sp로 설정한 후 return address를 push한다.

2. 함수 본체(compound statement)에 대해 generateCode를 호출한다.

3. return address를 pop한 뒤 fp를 pop 한다.

* generateCompound는 선언부와 statement로 나뉘는데, 선언부에서는 local variable의 크기를 구한 뒤, 그 만큼 스택 공간을 확보해둔다. 그 후에 generateCode를 호출하여 statement들에 대한 코드를 생성 한 후, stack pointer를 증가시켜 local variable을 해제하는 코드를 생성한다.
* generateReturn의 경우, sp를 fp-4로 설정 한 후, return address와 frame pointer를 pop 해주고, ra의 주소로 점프하도록 하였다.
* 함수 호출은 generateParamPassing을 호출하여 parameter를 pass 한 후, jal로 함수를 호출하고, sp를 증가시켜 pass한 parameter를 정리한다.
* generateParamPassing은 Call 할 때 넘겨줄 parameter들을 역순으로 계산하여 스택에 넣는 작업을 한다. 이 파라미터들은 함수 내부에서 fp + offset으로 참조되는데, 역순으로 넣었으므로 왼쪽부터 오른쪽으로 갈수록 offset이 증가한다.   
  code.c의 함수들은 각 label 및 instruction을 코드와 화면에 출력하는 역할을 한다. 이 때, 바로 fprintf를 쓰지 않고 이 함수들을 이용한 이유는 생성된 코드를 일괄적으로 정렬하여 관리하고 출력하기 위해서이다.

3. **제작 내용**:

(1) main()에서 CodeGen()을 호출한다.

->codeGen()은 generateInOut()을 호출하여 input과 output함수를 만든다.

->codeGen()은 이어서 generateCode()를 호출하여 코드를 생성한다.

(2) generateCode()는 node를 입력으로 받아 switch문으로 node의 종류에 따라

generateReturn(), generateFuncCall(), generateCompound(), generateIf(),

generateAssign(), generateConst(), generateRelop(), generateArrayVal(),

generateArrayDecl(), generateVal(), generateFunction(), generateValDecl()

generateArithmetic(), generateWhile()을 호출한다.

->generateFuncCall()은 generateParamPassing()을 호출하여

stack을 통한 parameter passing을 수행한다.

(3) code.c에는

emitPush(), emitPop(), [스택에 레지스터 값을 push하거나 스택에서 pop하는 기능]

instType1(), instType2(), instType3(), instType4(),

emitString(), emitArrayDecl(), emitLabel(),...

[string이나 전역변수 및 array, label, comment 등의 코드를 생성]

emitSyscall(), emitAddi(), emitLi(), ...

[각 instruction을 만듬. instType함수들과 printComment함수를 이용하여 구현]

printComment()

[fprintf(code,...)나 fprintf(listing,...)로 comment를 처리하기 위함]

가 있다.

이 함수들은 utility함수들로 cgen.c에서 코드를 생성할 때 사용하는 함수이다.

4. **시험** 내용:

프로젝트 4를 자체 평가하기 위해 조교님께서 올려주신 SPIM 예제 두 가지 ex1.c, ex2.c로 ./project4\_7 ex1[ex2].c command로 ex1.s와 ex2.s를 만들어 SPIM 머신에서 불러와 run하여 console창에 input을 넣거나 output을 출력받는다. 그 출력 값들이 예제와 같이 나왔다.

출제자가 직접 올려준 예제와 그 솔루션이기 때문에 그 예제와 같은 값이 나왔다는 것은 기본적인 구현은 완벽히 끝났다고 볼 수 있다.

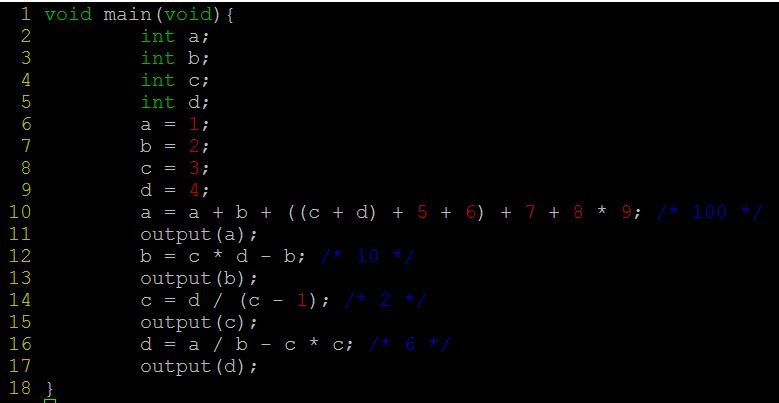
그 외에 자체 작성한 test1.c와 test2.c를 ./project4\_7 test1[test2].c command로 test1.s와 test2.s를 만들어 SPIM 머신에서 불러와 run하여 console창에 input을 넣거나 output을 출력받는다. 값들의 정답 여부는 c언어로 linux based의 putty에서 test\_1.c를 만들어 stdio.h를 참조하여 input은 scanf로 바꾸고 output은 printf로 바꾸어 적절히 c언어에 맞게 코드를 변경하여 나온 출력과 SPIM으로 .s파일을 run한 console 출력을 비교한다.

test1.c와 test2.c 모두 SPIM에서의 결과가 올바르게 나왔으므로 code generator의 구현이 잘 되었다고 볼 수 있다.

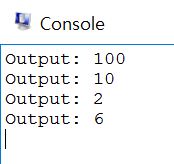
-test1.c와 test2.c 모두 올바른 input을 받는 한 같은 기능을 여러번 실행해볼 수 있다.

프로그램 예시 테스트

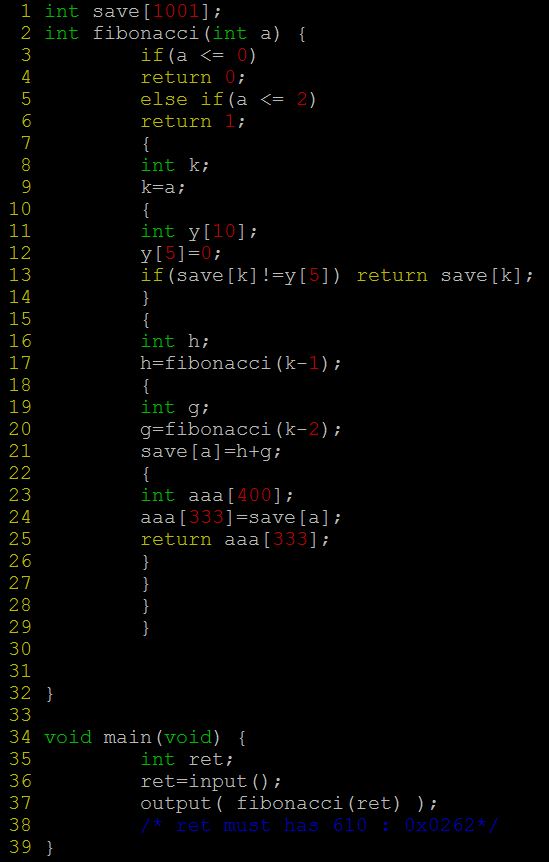
=========================ex1.c==============================



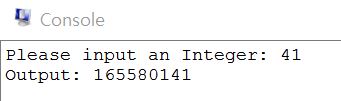
========================console with ex1.s=====================



=========================ex2.c==============================

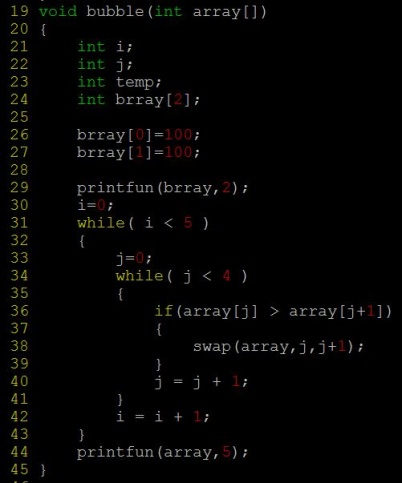
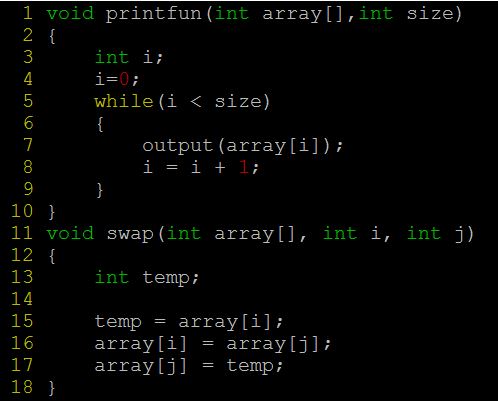


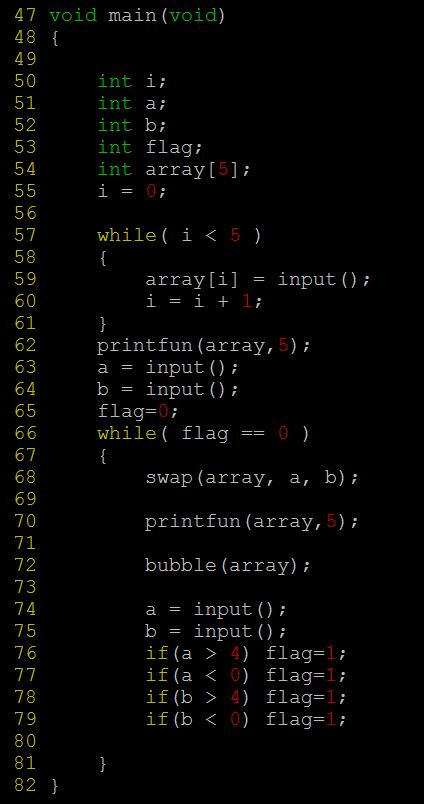
========================console with ex2.s=====================



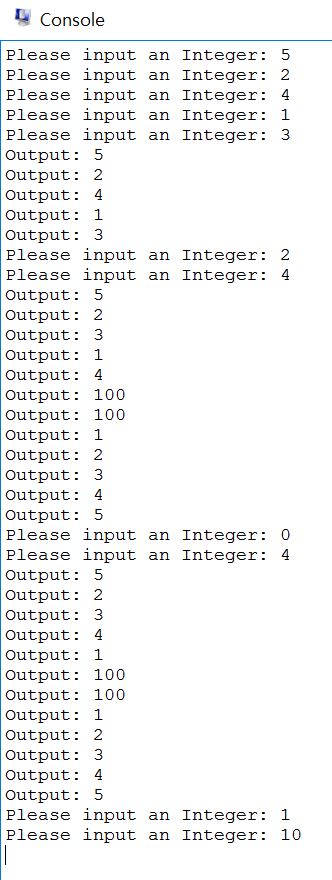
프로그램 자체 테스트

=======================test1.c==============================





======================console with test1.s=====================

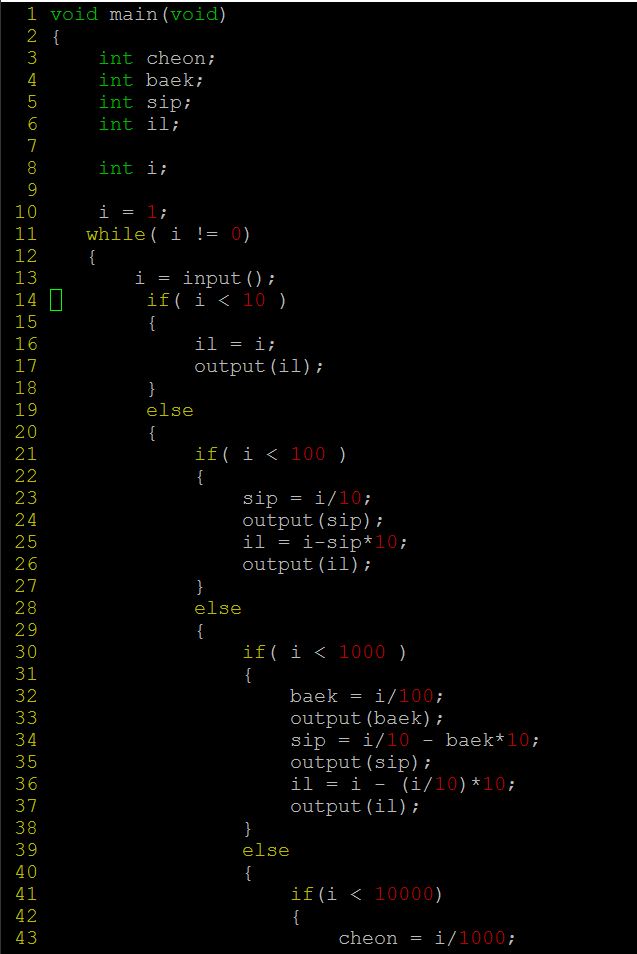


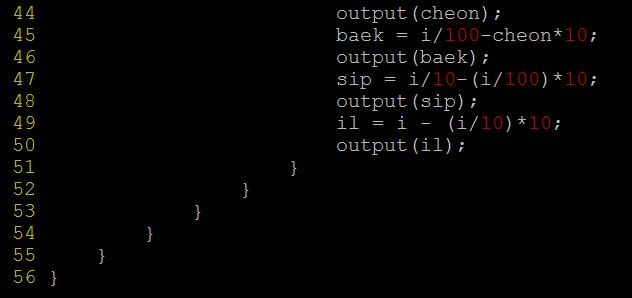
사용자에게 5개의 input을 차례로 배열에 받아 배열을 한번 출력해주고  
배열의 원소들의 순서를 swap하고 싶은 index를 두 정수로 입력 받습니다.

그러면 swap된 배열을 출력해준 뒤 100 100을 출력하고  
bubble sort된 배열을 재 출력해줍니다.

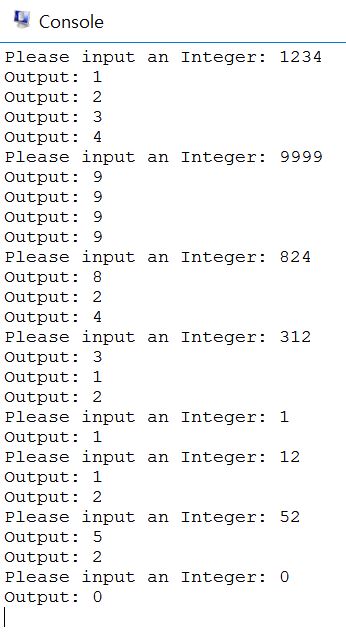
swap하고 싶은 index 두 정수가 valid(0~4)할 때까지 swap하고 sort하는 과정을 반복합니다.

=======================test2.c==============================





======================console with test2.s=====================



최대 4자리의 양의 정수를 입력받아 그 정수의 각 자릿수를 차례대로 출력해 준다.

해당 정수가 valid할 때 까지 계속한다.

5. **평가** 내용:

-장점: symbol table을 탐색하지 않고 AST의 필드에 필요한 정보를 직접 저장했으므로, 실행 속도가 빠르다. 또한 출력될 코드를 잘 정렬하여 보기 좋게 하였다.

-진행 도중 문제: test case중 main이 아닌 함수 A에서 배열을 인자로 갖는

다른 함수 B를 호출할 때 B에 넘겨줄 배열이 함수 A에서 선언한 배열일 때에는

pointer 1/2 pointer 2/2과정이고 함수 A에 인자로 넘어온 배열일 때에는

local address로 두 경우의 처리방식이 다르다. 그 부분을 처리해 주는 부분을

우리 프로젝트의 기존의 자료구조로는 구분할 수 있는 방법이 명확하지 않아

이 부분을 고치는 데에 고민을 많이 했다.

-해결 방법: 두 부분을 구분할 때에 symtab.c에 st\_getIsParam함수를 만들고

globals.h에 정의된 treeNode 구조체에 isParam이라는 변수를 추가하여

analyze.c에서 checkNode할 때에 isParam을 설정할 수 있도록 하여

이 문제를 해결했다. [자료구조를 조정함으로써 해결]

**경제 요건**: symbol table을 탐색하지 않고 AST를 바로 탐색하여 code generation하여 보다 빠른 시간 안에 문제를 해결했으므로 경제적이고 효율적이라고 할 수 있다.

**윤리성**: 컴퓨터 학과 학생들이 주로 범하는 실수인 copy를 하지 않고 스스로 처음부터 끝까지 구현하는 데에 힘썼기에 이 프로젝트는 윤리성 측면에서 좋은 프로젝트라고 할 수 있다.

**사회적 영향**: 4단계에 걸쳐 프로젝트를 진행하여 compiler를 이해할 수 있도록 하여 사회에 좋은 인재를 배출하는 좋은 영향을 미치는 프로젝트라고 할 수 있다.

**실행 가능성**: C- 에 맞는 source code를 compile하여 SPIM machine에서 실행 가능한 target code를 만드는 작업을 여러 테스트를 통해 증명했으므로 실행 가능성이 좋은 프로젝트라고 할 수 있다.

**안정성:** 본 팀 프로젝트의 결과 코드는 다양한 테스트 케이스에 대해, 기대되는

결과를 정확하게 나타낸다. 여러가지 오류 상황에 대하여 오류 내용을 정확하게

분석하여 사용자로 하여금 에러의 원인을 결과를 통해 확인할 수 있다. 이에 따라

안정성이 높은 프로그램이라고 볼 수 있다. 특히 abeek에 공지된 두 개의 예제

파일에 대해 에러 없이 수행하며 올바른 결과를 출력하는 것을 확인했다.

**신뢰성:** 똑같은 testcase에 대해 여러 번의 수행에도 같은 결과를 나타낸다면 프로그램은 신뢰성이 높다고 볼 수 있다. 본 프로젝트는 각종 testcase에 대해 10번 이상 수행하여도 결과는 같으며 segmentation error 등의 오류를 모두 제거하여 신뢰성 있는 프로그램이라고 볼 수 있다.

**V. 기타**

1. **연구 조원 기여도**:

|  |  |  |  |
| --- | --- | --- | --- |
| 조원 | 고원섭 | 이서희 | 정다은 |
| 비율 | 35.0% | 35.0% | 30.0% |

2. **자체 평가**:

수행 중 발생한 문제를 해결하기 위해 팀원들이 미팅을 통해 각자 의견을 내어 각 방법의 타당성을 서로 평가하고 장단점과 효율성을 도출하여 가장 적합한 의견으로 문제를 해결했다.

팀원 간의 협력이 조직적으로 이루어졌음을 보여주는 별첨 파일을 함께 제출합니다.

파일 이름: [Compiler]project\_4\_appendix\_7

3. 기타

**느낀 점**:

4번에 걸친 프로젝트를 수행함으로써 하나의 컴파일러를 만들어 보았다. 컴파일러의 작동 원리를 이해할 수 있었기 때문에 좋은 경험이었다.

SPIM 머신을 사용할 기회가 쉽게 주어지는 기회는 아닌데 졸업 전에 컴파일러 과목을 수강함으로서 좋은 경험을 할 수 있었다.

또한 팀을 이루어 진행하는 것은 혼자서 프로젝트를 하는 것에 비해 장점도 많고 단점도 많다. 그러나 팀을 이루는 것을 선택할 수 있도록 하는 것보다 모두 필수로 2인 이상으로 구성된 팀을 이루어 프로젝트를 진행하도록 하는 것이 팀으로 진행하는 프로젝트에 대해 느끼고 배우는 바가 더 크다고 생각한다.

* 참고 문헌

1. TextBook - Kenneth C. Louden, Compiler Construction: Principles and Practice, PWS Publishing Company, 1997.
2. 강의 자료
3. 위험 요소: 프로그래밍의 구현을 공동으로 작업하는 동안 적절하고 공평한 역할분담이 이루어 지도록 해야 한다.