1. **교재의 8장 pp.384-387의 연습문제 중 다음의 문제를 푸시오.**

**Review questions:**

1. **Define selection statement**

Selection statement은 대부분 명령형 언어에서 Algol 60때 소개된 if-then-else 개념을 기반으로 사용한다. 선택문의 기본 형태는 다음과 같다.

if condition then statement

else if condition then statement

else if condition then statement

…

else statement

그러나 Algol 60과 Pascal에선 아래와 같이 then과 else가 단일한 상태만을 가지게끔 정의되어서 이와 같은 형태로는 모호성 문제로 중첩 if문을 사용할 수 없다. 대표적인 예로 아래와 같은 경우를 들 수 있다. Condition2가 true 혹은 false이면 statement2를 실행할 지, statement3을 실행할지를 컴파일러에서 구분하지 못한다.

If condition1 then

If condition2 then statement1

else statement2

…

statement3

그래서 Algol 60의 경우 :begin과 같은 선언을 중첩문 시작 전에 작성해줘서 구별해주거나 Pascal의 경우 사용불가 처리를 하여 모호성을 해결한다. 또한 C언어의 경우 statement list가 구조의 끝에 종료 킹뤄드와 함께 then 혹은 else를 따르도록 한다. terminators들이 중첩의 끝에 쌓이는 것을 막기 위해, 대부분의 언어에서 elif elsif와 같은 키워드를 제공한다.

**3. What is the general form of a two-way selector?**

if (boolean\_expr)

then statement (the then clause)

else statement (the else clasue)

**10. Between what two language characteristics is a trade-off made when deciding whether more than one selectable segment is executed in one execution of a multiple selection statement?**

Multiple selectors의 경우 안정성과 유연성 간의 균형을 유지해야 한다.

**11. What is the role of the default segment in a switch statement?**

default segment는 switch statement에서 default label을 사용하여 선언되며, 이 레이블 아래의 코드는 switch expression과 일치하는 case가 존재하지 않을 경우 실행된다. default label은 선택사항이며 switch문당 하나만 존재한다. 일반적으로 switch문 블록의 마지막 label로 선언되지만 반드시 필요한 것은 아니다.

**13. Explain how C#’s switch statement is safer than that of C.**

C#은 둘 이상의 segment에 대한 암묵적 실행을 허용하지 않는 static semantics rule을 가지고 있다. 모든 segment는 switch문에서 제어를 전달하는 명시적 무조건 branch statement 혹은 선택 가능한 segment중 하나로 제어를 전달할 수 있는 goto로 끝나야 한다.

**17. What is the difference between a pretest version and a posttest version of a logical loop?**

사전 테스트의 경우 매번 루프조건을 테스트하고 조건식이 참 값인 동안에 계속 실행한다. 그러나 사후 테스트의 경우 실행 후에 하단의 루프 조건을 테스트하여 조건식이 참 값일 동안 반복한다.

**21. What are the design issues for logically controlled loop statements?**

사전 혹은 사후 테스트에 제어하는지에 대한 여부와 논리적으로 제어되는 루프는 특별한 형태의 카운팅 루프인지 아니면 별도의 명령문이 되어야 하는지에 대한 여부이다.

**22. What is the main reason user-located loop control statements were invented?**

사용자 임의의 루프 제어는 특정 경우에 반복문을 나가거나, 이하를 생략 후 다시 반복하라는 것들을 정의해 준다. C언어의 경우에는 break와 continue와 같은 구문이 존재한다. 이러한 사용자 임의의 루프 제어는 프로그램 동작 흐름에 있어서 프로그래머에게 편리함을 제공하며, 복잡한 프로그램 동작을 제공할 수 있다.

**25. What are the differences between the break statement of C++ and that of Java?**

C++에는 break 범위가 존재하는 루프만 중단할 수 있지만, JAVA는 대상 루프를 곧바로 중단 할 수 있다.

**Problem set:**

**9. What are the arguments both for and against the exclusive use of Boolean expressions in the control statements in Java (as opposed to also allowing arithmetic expressions, as in C++)?**

자바에서 Boolean expression만을 사용하면 구문의 규모가 더 커졌을 때 방해가 된다는 단점이 있다. 제어문에서 arithmetic expression을 허용한다면 다른 구현에 있어서 C++의 방식과 더욱 동일해지고 어떠한 구현에 있어서도 경쟁력이 있을 것이다. 그러나 컴파일러에 의해 오류로 감지되지 않는 타이핑 오류가 포함된다면 혼동을 일으킬 확률이 크다는 문제가 있다.

**12. What are the differences between the break statement of C++ and that of Java?**

goto문은 루프문장을 벗어날 때 주로 사용된다. 이는 반복문이 하나 이상 중첩된 경우 가장 바깥쪽 반복문까지 한 번에 탈출할 수 있다는 장점이 있다. goto문은 좀 더 빠른 기계어로 변환된다는 장점이 있다.

**Programming exercise:**

1. **a) Fortran 95**

integer :: k = (j + 13)/27

do while (k < 10)

k = k +1

I = 3 \* k – 1

end do

**b) Ada**

k := (j + 13)/27;

loop

k := k+1;

i := 3\*k-1;

exit when k = 10;

end loop;

**c) C, C++, Java or C#**

int k = (j + 13)/27;

do{

k = k+1;

i = 3\*k-1;

}while(k<10);

**d) Python**

k= (j+13)/27

while True:

k = k+1

i = 3\*k-1

if k>10: break

**e) Ruby**

k= (j+13)/27

until k > 10 [do]

k = k+1

i = 3 \* k -1

end

1. **C-Based**

for(k=((j+13)/27)+1.2);k<=10;k=k+1.2) {

i=3\*k-1;

}

**Python**

for k in range((((j+13)/27+1),11,0.2)

i=3\*k-1

**Ruby**

for k in (((j+13)/27)+1.2) .. 10 do

i=3\*k-1

k=k+0.2

end

**5.** boolean found = false;

for (i = 1, i <= n; i++) {

int counter = 0;

for (j = 1; j <= n; j++) {

if (x[i][j] == 0)

counter++;

}

if (counter == n && found == false) {

printf(“First all­zero row is: %d”, i);

found = true;

}

}

1. 8장의 동영상 강의 내용중 stepwize program development 에서 소개된 n개의 소수를 출력하는 프로그램의 version-3, version-4, version-5 및 과제 #5의 본인의 프로그램을 (실행시간을 나타내는 그래프를 보이고) 비교 분석하시오.

* 소스코드

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | #include <stdio.h> | | #include <stdlib.h> | | #include <unistd.h> | | #include <stdbool.h> | | #include <math.h> | | #include <string.h> | | #include <sys/time.h> | |  | | #define SECOND\_TO\_MICRO 1000000 | |  | | void version\_3(int input); | | void version\_4(int input); | | void version\_5(int input); | | void assign5(int input); | | void ssu\_runtime(struct timeval \*begin\_t, struct timeval \*end\_t); | | struct timeval begin\_t, end\_t; | |  | | int main(void) | | { | | int input; | | int num; // 소수 판별 대상 | | int denom; // 나눌 수 | | struct timeval begin\_t; | | struct timeval end\_t; | |  | | while (1) { | | printf("Input number of prime numbers(exit: -1): "); | | scanf("%d", &input); | | if(input < 0) | | break; | | assign5(input); | | version\_3(input); | | version\_4(input); | | version\_5(input); | | } | | } | |  | | /\*\* | | \* @brief 다음 소수를 구할 때 n-1까지의 모든 수 대입 | | \*/ | | void version\_3(int input) | | { | | int op = 3; | | int \*prime = (int \*)malloc(sizeof(int) \* input); | | int prime\_count = 0; | | bool is\_prime; | |  | | gettimeofday(&begin\_t, NULL); // 측정 시작 | |  | | prime[prime\_count++] = 2; | | while (prime\_count <= input) { | |  | | is\_prime = true; // 소수 확인 변수 초기화 | |  | | for (int i = 2; i < op; i++) // 최적 소수 인덱스 범위까지만 비교 | | if (op % i == 0) { // 소수로 나누어 떨어지는 경우 == 합성수 | | is\_prime = false; | | break; | | } | |  | | if (is\_prime) // 나누어 떨어지는 경우가 없음 == 소수 | | prime[prime\_count++] = op; | |  | | op += 2; // 홀수만 탐색 | | } | |  | | gettimeofday(&end\_t, NULL); // 측정 종료 | | #ifdef DEBUG | | for (int i = 0; i < input; i++) | | printf("%-d ", prime[i]); | | #endif | | printf("\nversion-3 complete\n"); | | ssu\_runtime(&begin\_t, &end\_t); | | free(prime); | | } | |  | | /\*\* | | \* @brief 다음 소수를 구할 때 n-1까지의 모든 소수 대입 | | \*/ | | void version\_4(int input) | | { | | int op = 3; | | int \*prime = (int \*)malloc(sizeof(int) \* input); | | int prime\_count = 0 ; | | bool is\_prime; | |  | | gettimeofday(&begin\_t, NULL); // 측정 시작 | |  | | prime[prime\_count++] = 2; | | while (prime\_count <= input) { | |  | | is\_prime = true; | |  | | for (int i = 0; i < prime\_count; i++) | | if (op % prime[i] == 0) { | | is\_prime = false; | | break; | | } | |  | | if (is\_prime) | | prime[prime\_count++] = op; | |  | | op += 2; | | } | |  | | gettimeofday(&end\_t, NULL); | | #ifdef DEBUG | | for(int i = 0; i < input; i++) | | printf("%-d ", prime[i]); | | #endif | | printf("\nversion-4 complete\n"); | | ssu\_runtime(&begin\_t, &end\_t); | | free(prime); | | } | |  | | /\*\* | | \* @brief 다음 소수를 구할 때 m까지의 소수로만 대입해보기 | | \*/ | | void version\_5(int input) | | { | | struct timeval begin\_t, end\_t; | | int op = 3; | | int limit\_idx = 1; // 최적 소수 인덱스 범위 | | int prime\_count = 0; // 소수 개수 | | int \*prime = (int \*)malloc(sizeof(int) \* input); | | bool is\_prime; // 소수 확인 변수 | |  | |  | | gettimeofday(&begin\_t, NULL); // 측정 시작 | |  | | prime[prime\_count++] = 2; // 소수 초기값 저장 | | while (prime\_count <= input) { // 소수 탐색 | |  | | is\_prime = true; // 소수 확인 변수 초기화 | |  | | if (sqrt(op) == prime[limit\_idx]) // 소수 판단 시 lim 범위 증가 | | limit\_idx++; | |  | | for (int i = 0; i < limit\_idx; i++) // 최적 소수 인덱스 범위까지만 비교 | | if (op % prime[i] == 0) { // 소수로 나누어 떨어지는 경우 == 합성수 | | is\_prime = false; | | break; | | } | |  | |  | | if (is\_prime) // 소수로 나누어 떨어지는 경우가 없음 == 소수 | | prime[prime\_count++] = op; | |  | | op += 2; // 홀수만 탐색 | | } | |  | | gettimeofday(&end\_t, NULL); // 측정 종료 | | #ifdef DEBUG | | for (int i = 0; i < input; i++) | | printf("%-d ", prime[i]); | | #endif | | printf("\nversion-5 complete\n"); | | ssu\_runtime(&begin\_t, &end\_t); | | free(prime); | | } | |  | | void assign5(int input) | | { | | int num; | | int denom; | |  | | gettimeofday(&begin\_t, NULL); // 측정 종료 | | num = denom = 2; | | for (int i = 0; i < input; i++) // 출력할 소수의 개수 만큼 반복 | | while (1) { | | if (num % denom == 0) // 나누었을 때 나머지가 0일 경우 | | if (num == denom) { // 소수일 경우 (자신 이전에 나눌 수 있는 수가 없음) | | #ifdef DEBUG | | printf("%-d ", denom); // 소수 출력 | | #endif | | num++; // 다음 수 탐색 | | denom = 2; // denom 초기화 | | break; // 다음 소수 탐색 | | } else { // 합성수일 경우 ( 자신 이전에 나눌 수 있는 수가 있음) | | num++; // 다음 수 탐색 | | denom = 2; // denom 초기화 | | } | | else denom++; // 모든 denom 탐색 | | } | |  | | gettimeofday(&end\_t, NULL); // 측정 종료 | | printf("\nassignment 5 algorithm complete\n"); | | ssu\_runtime(&begin\_t, &end\_t); | |  | | } | |  | | void ssu\_runtime(struct timeval \*begin\_t, struct timeval \*end\_t) // 실행시간 출력 | | { | | end\_t->tv\_sec -= begin\_t->tv\_sec; | |  | | if(end\_t->tv\_usec < begin\_t->tv\_usec){ | | end\_t->tv\_sec--; | | end\_t->tv\_usec += SECOND\_TO\_MICRO; | | } | |  | | end\_t->tv\_usec -= begin\_t->tv\_usec; | | printf("Runtime: %ld:%06ld(sec:usec)\n", end\_t->tv\_sec, end\_t->tv\_usec); | | } | |