**프로그래밍언어(가)**

**과제 6**

**이름 : 허예림**

**학번 : 20182667**

**교수님 : 유재우 교수님**

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

**Review questions**

**1) Define selection statement.**

선택문은 프로그램에서 둘 이상의 실행 경로 중에서 선택하는 수단을 제공한다. 2가지에는 if 문이 있으며 2가지 이상에는 case문, switch문이 있다.

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

if control\_expression, the clause, else clause

**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?**

Ada에서 case문의 선택문은 철저해야하므로 제어 표현식에 표시되지 않은 값이 없어야 한다. C++에서는 기본 선택으로 런타임에 표시되지 않은 값을 찾아낼 수 있다. 기본값이 없으면 표현되지 않은 값으로 인해 전체 명령문이 생략된다.

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

default은 switch문에 해당하는 값이 case문의 어떠한 경우에도 해당되지 않을 때 실행된다. switch문은 선택적으로 default의 경우를 가질 수 있으며, switch문의 끝에 표시되어야 한다. default문에서는 break가 필요하지 않다.

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

C# 스위치 문구는 두가지 면에서 C에 기반을 둔 이전 문들과 다르다. 첫째, c#는 둘 이상의 세그먼트의 암묵적 실행을 허용하지 않는 정적 의미론 규칙을 가지고 있다. 규칙은 모든 선택 가능한 세그먼트를 명시적인 무조건적인 분기 문장으로 끝나야 한다는 것이다. 즉, 스위치 문에서 제어권을 이전하는 휴식 또는 선택 가능한 세그먼트 중 하나로 제어권을 이전할 수 있는 goto가 있다. C와 마찬가지로 선택한 세그먼트 끝에 브레이크가 없으면 다음 세그먼트로 실행이 계속된다.

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

사전 테스트 루프(pretest version)는 먼저 테스트 표현식이 참인지 확인한 다음 루프 본문 실행을 계속한다. 사전 테스트 루프의 예시로는 for, while문이 있다. 테스트 후 루프(posttest version)는 먼저 루프의 본문을 적어도 한 번 실행한 다음 다시 실행하기 전에 표현식이 참인지 확인하는 것이다. 테스트 후 루프의 예시로는 do-while문이 있다.

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

디자인 문제는 컨트롤이 pretest, posttest 둘 중에 무엇이여야 하는지 여부와 논리적으로 제어되는 루프가 계산 루프의 특수한 형식이어야 하는지 또는 별도의 명령문이어야 하는지이다.

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

사용자 위치 루프 제어문이 발명된 주된 이유는 루프 본체의 상단 또는 하단 이외의 루프 제어를 위한 위치를 선택하기 위함이다.

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

C++에서 break은 두 가지 용도로만 사용된다. 첫 번째로는 switch문에서 case 실행을 종료하는데 사용된다. 두 번째로는 루프를 종료하고 루프 다음의 명령문으로 제어를 재개할 때 사용된다.

반면, Java의 break에는 먼저 세 가지 용도가 있다. 첫 번째로는 switch문에서 case를 종료하는데 사용된다. 두 번째로는 루프를 둘러싸는 break를 종료하고 , 세 번째로는 goto문을 제공하지 않지만 goto대신 확장된 형식을 사용할 수 있다.

즉, C++에는 이름 구분이있는, 무조건 종료 레이블이없는 종료가 있고 Java에는 이름이 같은, 무조건 레이블이 있는 종료가 있다. C++은 브레이크 범위가있는 루프만 중단 할 수 있지만 Java는 대상 루프로 곧바로 중단 될 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++)?**

부울 표현식을 제어 표현식으로 독점적으로 사용한다는 주요 주장은 이 유형에 대해 광범위한 유형을 허용하지 않기 때문에 발생하는 신뢰성 문제이다. 예를 들어, C에서 모든 유형의 표현식은 제어 표현식으로 나타날 수 있기 때문에 잘못된 유형의 변수에 대한 참조를 초래하는 입력 오류는 컴파일러에서 오류로 감지되지 않는다.

**12) Speculate as to the reason control can be transferred into a C loop statement.**

첫 번째 표현식은 초기화 용이며, for문이 실행될 때 한 번만 평가된다. 두 번째 표현식은 루프 제어이며, 루프 본문이 실행될 때마다 평간된다. C에서와 마찬가지로 0 값은 false를 의미하고 0이 아닌 모든 값은 true를 의미한다. 따라서, 두 번째 표현식의 값이 0이면 for문이 종료된다. 그렇지 않으면 루프 본문이 실행된다. C99에서 표현식은 부울 형식일 수도 있었다. C99 부울 형식은 값 0또는 1만 저장한다. for의 마지막 표현식은 루프 본문이 실행될 때마다 실행된다. 루프 카운트를 증가시키는 데 종종 사용된다. C에서 for문에 대한 운영 의미론적 설명이 표시된다.

**Programming exercises**

**1) Rewrite the following pseudocode segment using a loop structure in the specified languages:**

**k = (j + 13) / 27**

**loop:**

**if k > 10 then goto out k=k+ 1  
 i = 3 \* k-1  
 goto loop**

**out: . . .**

**a. C, C++, Java, or C#**

int k = (j+13)/27;

do {

k = k+1;

i = 3\*k-1;

}while(k < 10);

**b. Python**

k = (j+13)/27

while True:

k = k+1

i = 3\*k-1

if k > 10: break

**c. Ruby**

k = (j + 13)/27

until k > 10 [do]

k = k+1

i = 3\*k -1

end

**2) Redo Programming Exercise 1, except this time make all the variables and constants floating-point type, and change the statement**

**k=k+ 1**

**to**

**k = k + 1.2**

**a. c-Based**

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

{

i = 3\*k -1;

}

**b. Python**

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

i = 3\*k -1

**c. Ruby**

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

i = 3\*k -1

k = k+0.2

end

**5) In a letter to the editor of *CACM*, Rubin (1987) uses the following code segment as evidence that the readability of some code with gotos is bet- ter than the equivalent code without gotos. This code finds the first row of an *n* by *n* integer matrix named x that has nothing but zero values.**

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

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

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

**goto reject;**

**println ('First all-zero row is:', i);**

**break;**

**reject:**

**}**

**Rewrite this code without gotos in one of the following languages: C, C++, Java, or C#. Compare the readability of your code to that of the example code.**

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 all-zero row is : %d”,i);

found = true;

}

}

**문제 2. 8장 동영상 강의 내용 중 stepwize program development에서 소개된 n개의 소수를 출력하는 프로그램의 version-3, version-4, version-5 및 과제 #5의 본인의 프로그램을 (실행시간을 나타내는 그래프를 보이고) 비교 분석하시오.**

**1) 소스 코드**

#include<stdio.h>

#include<math.h>

#include<time.h>

#include<stdlib.h>

#define N 4

#pragma warning (disable:4996)

void primenum(int n);

void version3(int n);

void version4(int n);

void version5(int n);

int main()

{

int input[N] = {100,1000,5000,10000};

for (int i = 0; i < N; ++i)

{

printf("==========N : %d==========\n", input[i]);

primenum(input[i]);

version3(input[i]);

version4(input[i]);

version5(input[i]);

printf("==========================\n");

}

}

void primenum(int n){

int primenum;//소수이면 1 아니면 0을 저장한다.

int checknum;//소수인지 비교하는 값

int i = 0;

clock\_t start, end;

srand((unsigned int)time(NULL));

start = clock();

checknum = 2;//1은 소수가 아니기 때문에 2부터 검사한다.

while(i < n){

primenum = 1;

for(int j = 2; j < checknum; j++){

if(checknum % j == 0){

primenum = 0;

break;

}

}

if(primenum == 1){ // 소수이다.

i++;

}

checknum++;

}

end = clock();//종료 시간 측정

printf("primenum(%d)'s time: ", n);

printf("%.3lfmsec\n", (double)end - start);

}

void version3(int n){

clock\_t start, end;

srand((unsigned int)time(NULL));

start = clock();

int number = 2;

int count = 1;

int prim;

while (1){

++number;

prim = 0; //소수라고 가정하고시작

for (int i = 2; i <= number-1; ++i){

if (number%i == 0) //사이에 나눠떨어지는게있으면

prim = 1; //소수 아님

}

if (prim==0){

count++;

}

if (count == n)

break;

}

end = clock();

printf("version3(%d)'s time: ", n);

printf("%.3lfmsec\n", (double)end - start);

}

void version4(int n){

int i, k, lim, x;

int prim;

int p[10001];

clock\_t start, end;

srand((unsigned int)time(NULL));

start = clock();

p[1] = 2;

x = 1;

lim = 1;

for (i = 2; i <= n; i++){

x = x + 2;

k = 2;

prim = 1;

while (prim && (k < lim))

{

if (p[k] == 0)

break;

prim = (x%p[k] != 0);

++k;

}

if (prim)

p[i] = x;

else

--i;

}

end = clock();

printf("version4(%d)'s time:", n);

printf("%.3lfmsec\n", (double)end - start);

}

void version5(int n){

int i, k, lim, x;

int prim;

int p[10001];

clock\_t start, end;

srand((unsigned int)time(NULL));

start = clock();

p[1] = 2;

x = 1;

lim = 1;

for (i = 2; i <= n; ++i){

x = x + 2;

if (sqrt(p[lim]) <= x)

{

lim++;

}

k = 2;

prim = 1;

while (prim && (k < lim))

{

if (p[k] == 0)

break;

prim = (x%p[k] != 0);

++k;

}

if (prim)

p[i] = x;

else

--i;

}

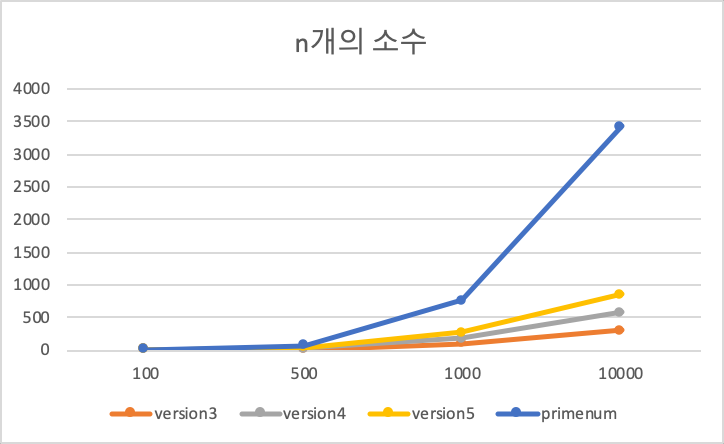
end = clock();

printf("version5(%d)'s time:", n);

printf("%.3lfmsec\n", (double)end - start);

}

**2) 비교 (그래프)**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 100 | 500 | 1000 | 10000 |
| primenum | 3 | 32 | 780 | 3400 |
| version3 | 1 | 11 | 238 | 379 |
| version4 | 2 | 9 | 280 | 621 |
| version5 | 1 | 12 | 390 | 839 |