데이터 구조

간단한 계산기 프로그램 프로그래밍

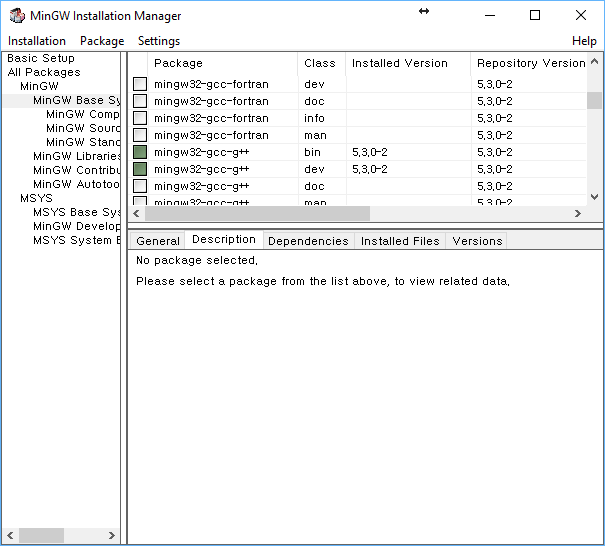
컴퓨터 공학

2015211373 권정인

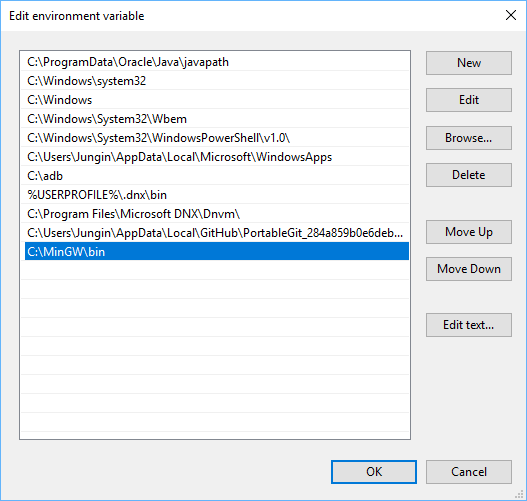
목차

1. 개요 및 작업 환경
2. 전체 코드
3. 컴파일 및 결과
4. 평가 및 개선 방안
5. 개요 및 작업 환경

저는 *Windows 10* 환경에서 코드를 작성하고, 이 코드를 빌드(컴파일)하기 위해 *GNU* 패키지에 있는 응용 프로그램인 *g++* 를 사용하였습니다. *Windows*용 *GNU* 패키지를 얻기 위해 *minGW* 를 사용 하였습니다. *minGW*를 설치 한 후, *g++* 패키지를 설치 한 후, 환경 변수(PATH)를 정의 하였습니다. (*minGW* 얻기 : <http://mingw.org/> )



*minGW* 패키지 관리 프로그램입니다. *Linux*의 *synaptic* 과 비슷한 사용자 인터페이스를 가지고 있습니다.



*minGW* 관련 바이너리 프로그램을 환경 변수에 추가하였습니다.

1. 전체 코드

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

#include <string.h>

#define MAXLEN 100

typedef struct StackNode

{

char data;

struct StackNode \*next;

}osNode;

typedef struct numStackNode

{

int num;

struct numStackNode \*next;

}nsNode;

typedef struct stackHead

{

int count;

osNode \*top;

}sHead;

typedef struct numStackHead

{

int count;

nsNode \*top;

}nsHead;

sHead createStack()

{

sHead sInit;

sInit.count=0;

sInit.top=NULL;

return sInit;

}

nsHead createNumStack()

{

nsHead nsInit;

nsInit.count=0;

nsInit.top=NULL;

return nsInit;

}

char process();

char in2post(char func[MAXLEN]);

int pushStack(sHead \*stack, char data);

int pushStack(nsHead \*stack, int integerData);

int popStack(sHead \*stack, char \*dataOut);

int popStack(nsHead \*stack, int \*operand);

int stackTop(sHead stack, char \*dataOut);

int emptyStack(sHead stack);

int priority(char token);

int fullStack(sHead stack);

int postFixEvaluate(char expr[MAXLEN]);

int calculate(int num1, char \_operator, int num2);

int destroyStack(sHead \*stack);

int destroyStack(nsHead \*stack);

int main(int argc, char \*argv[])

{

char character=NULL;

printf("\nWelcome to Calculator Micro Application\n");

printf("Usage Example : Input 3\*7 and Press [Enter]\n");

printf("Input Q or q to halt\n\n");

while(character!='q' && character!='Q')

{

character=process();

}

printf("Halting Calculator Application..\n\n");

return 0;

}

char process()

{

char func[MAXLEN]={'\0',};

fflush(stdin);

gets(func);

if(func[0]=='Q')

return 'Q';

else if(func[0]=='q')

return 'q';

in2post(func);

return NULL;

}

char in2post(char func[MAXLEN])

{

sHead stack=createStack();

char postFix[MAXLEN]={'\0',};

char token;

char topToken;

char tokenOut;

int formulaLen = strlen(func);

int looper=0;

int index=0;

while(looper < formulaLen)

{

token = func[looper];

if(token == '(')

pushStack(&stack,token);

else if(token == ')')

{

popStack(&stack,&token);

while(token!='(')

{

postFix[index]=token;

index++;

popStack(&stack,&token);

}

}

else if(token == '+' || token == '-' || token == '\*' || token == '/')

{

stackTop(stack,&topToken);

while(!emptyStack(stack) && priority(token) <= priority(topToken))

{

popStack(&stack,&tokenOut);

postFix[index]=tokenOut;

index++;

stackTop(stack,&topToken);

}

pushStack(&stack,token);

}

else

{

postFix[index]=token;

index++;

}

looper = looper + 1;

}

while(!emptyStack(stack))

{

popStack(&stack,&token);

postFix[index]=token;

index++;

}

destroyStack(&stack);

printf("=%d\n",postFixEvaluate(postFix));

return NULL;

}

int pushStack(sHead \*stack, char data)

{

osNode \*newPtr=NULL;

int success;

if(!(newPtr = (osNode \*)malloc(sizeof(osNode))))

success = 0;

else

{

newPtr->data = data;

newPtr->next = stack->top;

stack->top = newPtr;

stack->count = stack->count+1;

success = 1;

}

return success;

}

int popStack(sHead \*stack, char \*dataOut)

{

osNode \*dltPtr=NULL;

int success;

if(stack->count == 0)

success = 0;

else

{

dltPtr = stack->top;

\*dataOut = stack->top->data;

stack->top = stack->top->next;

stack->count = stack->count-1;

free(dltPtr);

success = 1;

}

return success;

}

int stackTop(sHead stack, char \*dataOut)

{

int success;

if(stack.count == 0)

success=0;

else

{

\*dataOut = stack.top->data;

success = 1;

}

return success;

}

int emptyStack(sHead stack)

{

int result;

if(stack.count != 0)

result = 0;

else

result = 1;

return result;

}

int priority(char token)

{

int order;

switch(token)

{

case '+':

order=1;

break;

case '-':

order=1;

break;

case '\*':

order=2;

break;

case '/':

order=2;

break;

default :

order=0;

break;

}

return order;

}

int destroyStack(sHead \*stack)

{

osNode \*temp=NULL;

while(stack->top!=NULL)

{

temp = stack->top;

stack->top = stack->top->next;

free(temp);

}

stack->count = 0;

return 1;

}

int fullStack(sHead stack)

{

osNode \*temp=NULL;

temp = (osNode \*)malloc(sizeof(osNode));

if(temp!=NULL)

{

free(temp);

return 0;

}

return 1;

}

int postFixEvaluate(char expr[MAXLEN])

{

int exprSize = strlen(expr);

nsHead stack=createNumStack();

int index=0;

int num1;

int num2;

char \_operator;

int value;

int result;

while(index < exprSize)

{

if(expr[index]!='+' && expr[index]!='-' && expr[index]!='\*' && expr[index]!='/')

pushStack(&stack,(expr[index]-'0'));

else

{

popStack(&stack,&num2);

popStack(&stack,&num1);

\_operator = expr[index];

value = calculate(num1,\_operator,num2);

pushStack(&stack,value);

}

index = index +1;

}

popStack(&stack,&result);

destroyStack(&stack);

return result;

}

int pushStack(nsHead \*stack, int IntegerData)

{

nsNode \*newPtr=NULL;

int success;

if(!(newPtr = (nsNode \*)malloc(sizeof(nsNode))))

success = 0;

else

{

newPtr->num = IntegerData;

newPtr->next = stack->top;

stack->top = newPtr;

stack->count = stack->count+1;

success=1;

}

return success;

}

int popStack(nsHead \*stack, int \*operand)

{

nsNode \*dltPtr=NULL;

int success;

if(stack->count==0)

success=0;

else

{

dltPtr = stack->top;

\*operand = stack->top->num;

stack->top = stack->top->next;

stack->count = stack->count-1;

free(dltPtr);

success = 1;

}

return success;

}

int calculate(int num1, char \_operator, int num2)

{

int result=0;

switch(\_operator)

{

case '+':

result = num1+num2;

break;

case '-':

result = num1-num2;

break;

case '\*':

result = num1\*num2;

break;

case '/':

result = num1/num2;

break;

}

return result;

}

int destroyStack(nsHead \*stack)

{

nsNode \*temp=NULL;

while(stack->top!=NULL)

{

temp = stack->top;

stack->top = stack->top->next;

free(temp);

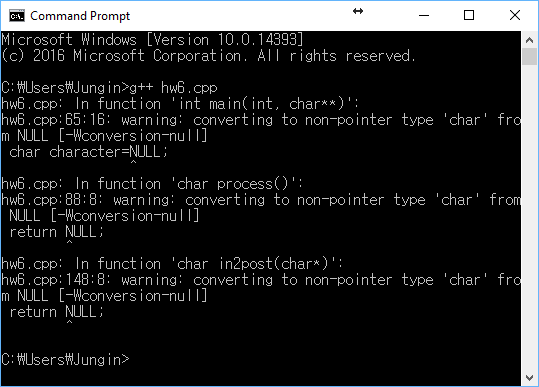
}

stack->count = 0;

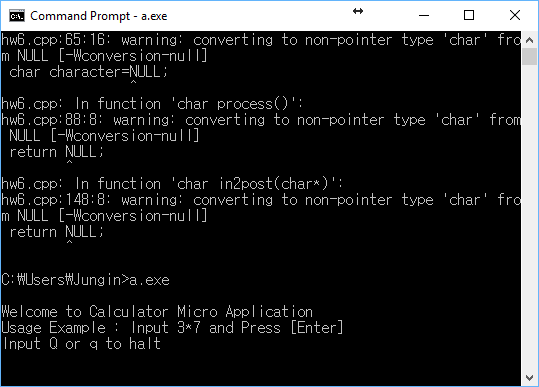
return 1;

}

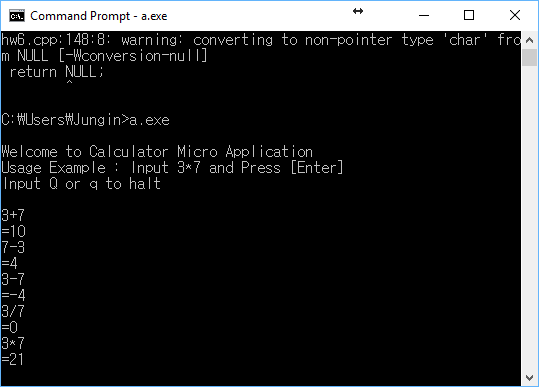
1. 컴파일 및 결과



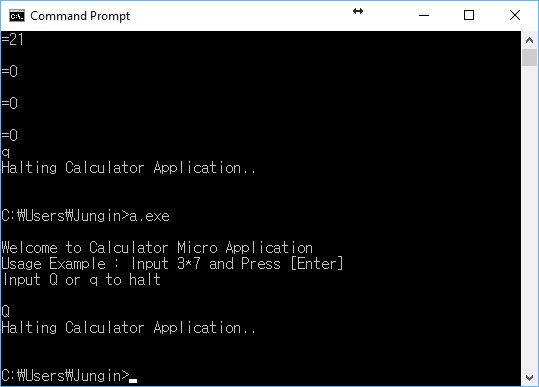
1. *Windows 10* 환경에서 *g++* 를 이용해 소스파일은 *hw6.cpp* 파일을 컴파일(빌드)합니다. [주의]사항의 발생하지만, 무시할 수 있는 항목입니다.



2. 빌드(컴파일)한 후 출력 된 a.exe 프로그램을 실행 하였습니다.



3. 실행한 후, 간단한 사칙연산을 계산해 보았습니다. 값이 음수라도 정상적으로 출력됨을 확인할 수 있었습니다. 단, 나눗셈에서 [나머지]값은 계산해 주지 않았습니다.



4. ‘q’ 또는 ‘Q’를 입력 받을 시 계산기 응용 프로그램이 정상적으로 종료됨을 확인할 수 있었습니다.

1. 평가 및 개선 방안
2. 과제 내용에서는 대문자 Q를 입력받으면 프로그램이 종료되도록 되어 있습니다. 하지만, 대부분의 키보드 상태에는 *CapsLock*이 꺼져 있다는점과, 사용자의 편의를 위하여 소문자 ‘q’를 입력 받을때도 응용 프로그램이 종료되도록 설계하였습니다.
3. 처음에는 모든 함수를 *main* 앞에 전부 첨부 하였습니다. 그랬더니, 컴파일 시 오류가 다수 발견 되었습니다. 학교 선배의 도움을 구하여, 함수 정의만 *main* 앞에 해주고, 함수 첨부는 그 후에 작성해야 함을 알았으며, 그렇게 했더니 정상적으로 컴파일이 됨을 확인할 수 있었습니다.
4. *Windows* 내에서 한글 출력이 정상적으로 되지 않습니다. 이 문제는 *Linux* 기반의 *Ubuntu* 라는 운영체제에서도 같은 문제가 발생하였습니다. 코드를 작성할 때 인코딩 오류인지, 아니면 컴파일하고 실행하는 과정에서 문제가 발생하였는지 모르겠습니다. 곧 수정하도록 노력 해 보겠습니다.
5. 계산기 프로그램에서 나눗셈 계산을 수행할 시, 몫으로 떨어지지 않을 경우(나머지 값이 있을 경우), 나머지도 동시에 표시하도록 프로그램을 설계할 수 있으면 더 좋을 것 같습니다