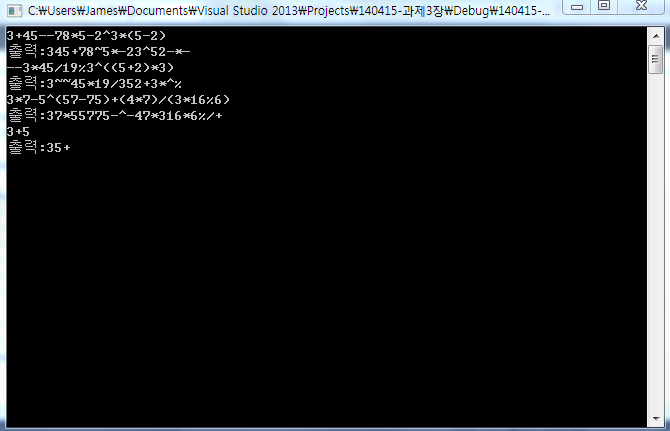


|  |
| --- |
| 음수 지원 계산기 |
| **2주차 핵심 연구 최종 결과(Infix->Postfix)로의 전환** |

|  |  |
| --- | --- |
| **이 름** | **정 O O** |
| **이메일** |  |
| **작성일** | **2014/04/19** |

**1. 출력결과**



**최종 실험 Framework 출력:**

**클라이언트 요구결과**

|  |
| --- |
| 계산식을 입력하세요 (종료는 quit) : 3+45 - - 78\*5 – 2^3 \* (5 -2)  --- 후위 계산식은 : 3 45 + 78 ~ 5 \* - 2 3 ^ 5 2 - \* -  ---- 결과는 414  계산식을 입력하세요 (종료는 quit) : --3 \* 45 / 19 % 3 ^ ((5+2)\*3)  --- 후위 계산식은 : 3 ~ ~ 45 \* 19 / 3 5 2 + 3 \* ^ %  ---- 결과는 7  계산식을 입력하세요 (종료는 quit) : 3 \* 7 – 5 ^ ( 57 - 75) + (4 \* 7) / (3 \* 16 % 6)  --- 후위 계산식은 : 3 7 \* 5 57 75 - ^ - 4 7 \* 3 16 \* 6 % / +  [오류] 나누기의 두번째 연산자가 0인 경우가 발생  계산식을 입력하세요 (종료는 quit) : 3 + \* 7 0  [오류] 이해할 수 없는 수식  계산식을 입력하세요 (종료는 quit) :quit  \*\*\*\* 종료합니다 \*\*\*\* |

**4. 코드**

|  |
| --- |
| #ifndef \_CALC\_H\_  #define \_CALC\_H\_  #include <string>  #include "stack.h"  using namespace std;  const int INFIX = 1;  const int POSTFIX = 2;  const int PARAENTHESE = 3;  template <class T>  class Calc{  public:  Calc(){};  ~Calc(){};  void SetSize(int \_Size){ Size = \_Size - 1; };  void GetPostfix(string \_GetData);  int GetPrevOpCount(string \_GetData, int \_PrevIDX, int \_LastIDX);  void PrintOut(int Type);  int ISP(char \_Data);  int ICP(char \_Data);  char StringToChar(int IDX, string \_GetData);  bool IsOpChk(char \_Data);  bool IsPrevParaenthese(string \_GetData, int \_PrevIDX, int LastIDX);    protected:  Stack<char>\* OpExp;  Stack<char>\* PostExp;  int Size;  }; |

Figure ) ADT 설계

|  |
| --- |
| template <class T>  void Calc<T>::GetPostfix(string \_GetData){  int IDX;  int PrevIDX;  int NextIDX;  int LastIDX;  char TmpData;  char PrevData;  char NextData;  bool Check;  PostExp = new Stack<char>();  OpExp = new Stack<char>();  Stack<char>\* TmpExp = new Stack<char>();  PostExp->SetStack(Size \* 2);  OpExp->SetStack(Size \* 2);  TmpExp->SetStack(Size \* 2);  IDX = 0;  PrevIDX = IDX;  Check = false; // 초기 문자가 아닐때 예상  while (IDX <= Size)  {  NextData = StringToChar(IDX, \_GetData);  // 숫자일 때  if (isdigit(NextData))  {  PostExp->Push(NextData);  Check = true;  }    // 문자일 때  if (!isdigit(NextData))  {  // 숫자 앞에 문자일 때  if (Check == false)  {  OpExp->Push(NextData);  if(NextData == '(')  PrevIDX = IDX;  IDX++;  continue;  }  // ')괄호일 때')  if (NextData == ')')  {  while (OpExp->GetTop() > -1)  {  PrevData = OpExp->GetData(OpExp->GetTop());    if (PrevData == '(')  {  NextData = OpExp->Pop();  PrevData = OpExp->GetData(OpExp->GetTop());  if (PrevData == '(' && NextData == '(')  break;  else  {  continue;  }  }  // 이전 (괄호일 때, 다음값 (괄호일 때  if (PrevData == '(' && NextData == '(')  {  break;  }  // ISP(NextData) <= ICP(PrevData)  if (ISP(NextData) <= ICP(PrevData))  {  if (NextData == '(' || PrevData == '(')  {  if (NextData == PrevData)  {  NextData = OpExp->Pop();  PostExp->Push(NextData);  }  else{  // 입력 범위가 맞는지 확인  if (NextData != '(' && PrevData != '^')  break;  else  {  // ^일 때  TmpData = PostExp->GetData(PostExp->GetTop());    if (TmpData == '^'){  NextData = OpExp->Pop();  PostExp->Push(NextData);  }  else  break;  }  }  }  else{  NextData = OpExp->Pop();  PostExp->Push(NextData);  }  }  else  {  PrevData = OpExp->Pop();  PostExp->Push(PrevData);  if (PrevData == '(' && NextData == '(')  break;  else  {  continue;  }  }  }  // 마지막 일 때  if (IDX == Size)  {  while (OpExp->GetTop() > -1)  {  PrevData = OpExp->Pop();  if (ISP(PrevData) == PARAENTHESE)  continue;  PostExp->Push(PrevData);  }  break;  }  PrevIDX = IDX;  IDX++;  Check = false;  continue;  }  if (Check == true){  LastIDX = IDX;  // 맨 처음 문자 앞문자가 -음수일 때  if (GetPrevOpCount(\_GetData, PrevIDX, LastIDX) > 0 && PrevIDX == 0)  {  while (OpExp->GetTop() != -1)  {  PrevData = OpExp->Pop();  if (PrevData == '-')  PostExp->Push('~');  }  OpExp->Push(NextData);  PrevIDX = IDX;  IDX++;  Check = false;  continue;  }  // 맨 처음 문자 앞문자가 -음수가 아닐 때  if (GetPrevOpCount(\_GetData, PrevIDX, LastIDX) == 0 && PrevIDX == 0)  {  OpExp->Push(NextData);  PrevIDX = IDX;  IDX++;  Check = false;  continue;  }  // ISP(NextData) < ICP(PrevData)가 참일 때  PrevData = StringToChar(PrevIDX, \_GetData);  if (ISP(NextData) < ICP(PrevData))  {  // PrevIDX(이전 범위부터 현재 범위까지 -음수갯수 2이상일 때  if (GetPrevOpCount(\_GetData, PrevIDX, IDX) > 1)  {  while (OpExp->GetTop() > 0)  {  PrevData = OpExp->GetData(OpExp->GetTop());  if (PrevData == '-'){  PrevData = OpExp->Pop();  PostExp->Push('~');  }  }  PrevIDX = IDX;  OpExp->Push(NextData);  Check = false;  IDX++;  continue;  }  else  {  // 승수가 아닐 때  if (NextData != '^')  {  while (OpExp->GetTop() > -1)  {  PrevData = OpExp->GetData(OpExp->GetTop());  if (PrevData == '(')  break;  else  {  PrevData = OpExp->Pop();  PostExp->Push(PrevData);  PrevData = StringToChar(PrevIDX, \_GetData);  if (PrevData == '^')  break;  }  }  PrevIDX = IDX;  OpExp->Push(NextData);  Check = false;  IDX++;  continue;  }  // ^일 때  else  {  OpExp->Push(NextData);    Check = false;  PrevIDX = IDX;  IDX++;  continue;  }  }  }  // ISP(NextData) >= ICP(PrevData)  else  {  while (OpExp->GetTop() > -1)  {  PrevData = OpExp->GetData(OpExp->GetTop());    // 이전값 보다 낮을 때  if (ISP(PrevData) < ICP(NextData))  break;  else  {  // 제곱근일 때  if (NextData == '^')  break;  // 괄호일 때  if (PrevData == '(')  break;  PrevData = OpExp->Pop();  // 괄호 생략  if (ISP(PrevData) == PARAENTHESE)  continue;  PostExp->Push(PrevData);  }  }    PrevIDX = IDX;  OpExp->Push(NextData);  Check = false;  IDX++;  continue;  }  }  else  {  PrevData = OpExp->Pop();  PostExp->Push(PrevData);  PrevIDX = IDX;  OpExp->Push(NextData);  Check = false;  IDX++;  }  }  // 마지막 일 때  if (IDX == Size)  {  while (OpExp->GetTop() > -1)  {  PrevData = OpExp->Pop();  if (ISP(PrevData) == PARAENTHESE)  continue;  PostExp->Push(PrevData);  }  }  IDX++;  }  } |

Figure ) 계산기 핵심(음수지원용)

|  |
| --- |
| template <class T>  int Calc<T>::GetPrevOpCount(string \_GetData, int \_PrevIDX, int \_LastIDX){  string GetData;  char TransData;  int PrevIDX = \_PrevIDX;  int LastIDX = \_LastIDX;  int Count = 0;  while (PrevIDX < LastIDX)  {  GetData = \_GetData.substr(PrevIDX, 1);  TransData = GetData.at(0);    if (TransData == '-')  Count++;  PrevIDX++;  }  return Count;  } |

Figure ) 이전 음수 부호 추적

|  |
| --- |
| template <class T>  int Calc<T>::ISP(char \_Data){  if (\_Data == '+' || \_Data == '-')  return 0;  if (\_Data == '%' || \_Data == '/' || \_Data == '\*')  return 1;  if (isdigit(\_Data) || \_Data == '^')  return 2;  if (\_Data == '(' || \_Data == ')')  return 3;  return -1;    }  template <class T>  int Calc<T>::ICP(char \_Data){  if (\_Data == '+' || \_Data == '-')  return 3;  if (\_Data == '%' || \_Data == '/' || \_Data == '\*')  return 2;  if (isdigit(\_Data) || \_Data == '^')  return 1;  if (\_Data == '(' || \_Data == ')')  return 0;  return -1;  } |

Figure ) ISP / ICP 핵심

|  |
| --- |
| template <class T>  char Calc<T>::StringToChar(int IDX, string \_GetData){  string GetData = \_GetData.substr(IDX, 1);  return GetData.at(0);  } |

Figure ) String To Char(문자 - char형식으로 변환)

|  |
| --- |
| template <class T>  void Calc<T>::PrintOut(int Type){  for (int i = 0; i <= PostExp->GetTop(); i++)  {  switch (Type)  {  case POSTFIX:  cout << PostExp->GetData(i);  break;  }  }  cout << endl;  }  #endif |

Figure ) 출력

**5. 진행 일정**-> 2014. 4. 8(착수) ~ 2014. 5. 6(화요일)마감)  
(2주차 진행중)

**6. 향후 진행 사항**

**- 계산 처리  
- 오류 판별** (연산자가 0일 때  
 (이해할 수 없는 수식 판별) **- 종료 구현  
  
고객의 나머지 요구사항**

|  |
| --- |
| 계산식을 입력하세요 (종료는 quit) : 3+45 - - 78\*5 – 2^3 \* (5 -2)  --- 후위 계산식은 : 3 45 + 78 ~ 5 \* - 2 3 ^ 5 2 - \* -  ---- 결과는 414  계산식을 입력하세요 (종료는 quit) : --3 \* 45 / 19 % 3 ^ ((5+2)\*3)  --- 후위 계산식은 : 3 ~ ~ 45 \* 19 / 3 5 2 + 3 \* ^ %  ---- 결과는 7  계산식을 입력하세요 (종료는 quit) : 3 \* 7 – 5 ^ ( 57 - 75) + (4 \* 7) / (3 \* 16 % 6)  --- 후위 계산식은 : 3 7 \* 5 57 75 - ^ - 4 7 \* 3 16 \* 6 % / +  [오류] 나누기의 두번째 연산자가 0인 경우가 발생  계산식을 입력하세요 (종료는 quit) : 3 + \* 7 0  [오류] 이해할 수 없는 수식 |

**일정 계획 25일~26일(수행 계획) – Gumi-City -> In Gwangju  
사유: 시험공부(진행)로 인함. (4월 19일 이후로 잠시 중단)**