UNIVERSIDAD NACIONAL SAN AGUSTIN DE AREQUIPA
FACULTAD DE INGENIERÍAS
ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN

LABORATORIO O6

Asignatura: Ciencia de la Computación II

Docente: Rolando Jesus Cardenas Talavera

Grupo: B

Estudiante: Roni Ezequiel Montañez Pacco

Arequipa-Perú

24 de mayo del 2024

Ejercicios

1.

```
~/2024A/ccii/Laboratorio/Laboratorio06  main !2 ?4 ❯ build/ejercicio1
Imprimiendo la clase Inferior:
Inferior
Imprimiendo la clase Superior desde un objeto de la clase Inferior:
Superior
```

CÓDIGO

```cpp
#include <iostream>
using namespace std;
class Superior {
  public:
    virtual void print(){
      cout << "Superior" << endl;
    }
};

class Inferior : public Superior{
  public:
    void print(){
      cout << "Inferior" << endl;
    }
};

int main(){
  Inferior infe;
  cout << "Imprimiendo la clase Inferior:\n";
  infe.print();
  cout << "Imprimiendo la clase Superior desde un objeto de la clase Inferior:\n";
  infe.Superior::print();
}
```

Se puede mediante ::, ya que la clase derivada hereda primero los metodos de la clase padre, es por ello que aunque redefinamos la funcion en la clase derivada, aun tendra el metodo de la clase superior.

2.

```
3+5*2-4/2^2
Resultado: 12
```

CÓDIGO

```cpp
#include <iostream>
using namespace std;
using namespace std;

double potencia(double bas,int exp){
  double res;
  if(exp==0){
    res = 1;
  }else{
    res = bas;
    int _exp = (exp < 0 ? exp*-1 : exp);
    for(int i=1; i<_exp; i++){
      res*=bas;
    }
    if(exp<0){
      res = 1/res;
    }
  }
  return res;
}

double convertir(const char *str){
  bool isint = true;
  double decimal=0;
  int itdec=-1;
  double signo = 1;
  while(*str != '\0'){
    if(*str=='-'){
      signo = -1;
    }else if(*str=='.'){
      isint = false;
    }else if(*str==' '){
      ;
    }else{
      if(isint==true){
        decimal *= 10;
        decimal += double(*str)-48;
      }else{
        decimal += (double(*str)-48)*potencia(10,itdec);
        itdec--;
      }
    }
    str++;
  }
  return decimal*signo;
}

int posminoroperator(const char *str){
  int pos=-1, it=0;
  char oper;
  while(str[it] != '\0'){
    switch(str[it]){
```

```
      case '-':
      case '+':
        if(it!=0 && (str[it-1]!='-' && str[it-1]!='*' && str[it-1] != '+' && str[it-1] != '^' && str[it-1] != '/
          return it;
        }
        break;
      case '/':
      case '*':
        pos = it;
        oper = str[it];
        break;
      case '%':
        if(oper != '/' && oper != '*'){
          pos = it;
          oper = str[it];
        }
        break;
      case '^':
        if(oper != '/' && oper != '*' && oper != '%'){
          pos = it;
        }
        break;
      }
      it++;
    }
  return pos;
}

class Nodo{
  public:
    virtual double operar() = 0;
    virtual ~Nodo () {}
};

class NumNodo : public Nodo{
    double num;
  public:
    NumNodo(double num) : num(num) {}
    double operar(){return num;}
};

class OperNodo : public Nodo{
  protected:
    Nodo *hizq;
    Nodo *hder;
  public:
    OperNodo(Nodo *hizq, Nodo *hder) : hizq(hizq), hder(hder) {}
    ~OperNodo(){delete hder; delete hizq;}
};

class SumaNodo : public OperNodo{
  public:
    SumaNodo(Nodo *hizq, Nodo *hder): OperNodo(hizq, hder) {}
```

```
    double operar(){return hizq->operar() + hder->operar();}
};

class RestaNodo : public OperNodo{
  public:
    RestaNodo(Nodo *hizq, Nodo *hder): OperNodo(hizq, hder) {}
    double operar(){return hizq->operar() - hder->operar();}
};

class MultNodo: public OperNodo{
  public:
    MultNodo(Nodo *hizq, Nodo *hder): OperNodo(hizq, hder) {}
    double operar(){return hizq->operar() * hder->operar();}
};

class DivNodo : public OperNodo{
  public:
    DivNodo(Nodo *hizq, Nodo *hder): OperNodo(hizq, hder) {}
    double operar(){return hizq->operar() / hder->operar();}
};

class PercentNodo : public OperNodo{
  public:
    PercentNodo(Nodo *hizq, Nodo *hder): OperNodo(hizq, hder) {}
    double operar(){return hizq->operar()/100 * hder->operar();}
};

class PotenNodo : public OperNodo{
  public:
    PotenNodo(Nodo *hizq, Nodo *hder): OperNodo(hizq, hder) {}
    double operar(){return potencia(hizq->operar(), hder->operar());}
};

int size(const char* str){
  int tam=0;
  for(int i=0; *str!='\0'; str++, tam++ )
    ;
  return tam;
}

char *cortar(const char *str, int ini, int fin){
  char *tmp= new char[fin-ini+1];
  for(int i=0; i<fin-ini+1; i++){
    tmp[i]=str[ini+i];
  }
  return tmp;
}
Nodo* resultado(const char* str){
  int tam=size(str);
  int posminor= posminoroperator(str);
  if(posminor == -1){
    return new NumNodo(convertir(str));
  }
```

```
  char *strizq = cortar(str,0,posminor-1);
  char *strder = cortar(str,posminor+1,tam-1);
  Nodo *hizq = resultado(strizq);
  Nodo *hder = resultado(strder);
  switch(str[posminor]){
    case '+':
      return new SumaNodo(hizq, hder);
    case '-':
      return new RestaNodo(hizq, hder);
    case '*':
      return new MultNodo(hizq, hder);
    case '/':
      return new DivNodo(hizq, hder);
    case '%':
      return new PercentNodo(hizq, hder);
    case '^':
      return new PotenNodo(hizq, hder);
    }
    return nullptr;
}

int main(){
  char *str = new char[100];
  char tmp;
  int it=0;
  while(cin.get(tmp)){
    if(tmp=='\n'){
      break;
    }
    str[it] = tmp;
    it++;
  }
  Nodo *result = resultado(str);
  cout << "Resultado: " << result->operar()<<endl;
}
```