Llistes amb accés i inserció per índex — X19702_ca

En aquest exercici heu d'implementar un programa que executa una seqüència de comandes d'entrada. D'entre aquestes comandes, n'hi ha que incrementen o decrementen una variable `index` que se suposa inicialitzada a 0.
A part de les comandes que modifiquen la variable `index`, n'hi ha d'altres que modifiquen o consulten una estructura de dades que és una mena de barreja entre llistes d'enters i vectors d'enters. Per una banda, hi han comandes per afegir elements al principi o al final. Per altra banda, hi han comandes que accedeixen indexadament als elements, i comandes per a insertar per índex nous elements. Això sí, totes aquestes comandes indexen sempre usant la variable `index` mencionada anteriorment.
Aquest és un exemple d'entrada del programa:

```
v.push_back( 5 );          // index == 0, v == [5]
cout<<v[index]<<endl;      // output: 5
v.push_front( 8 );         // index == 0, v == [8, 5]
cout<<v[index]<<endl;      // output: 8
index++;                   // index == 1, v == [8, 5]
cout<<v[index]<<endl;      // output: 5
v.push_back( 1 );          // index == 1, v == [8, 5, 1]
cout<<v[index]<<endl;      // output: 5
index--;                   // index == 0, v == [8, 5, 1]
cout<<v[index]<<endl;      // output: 8
v.insert(index, 4 );       // index == 0, v == [4, 8, 5, 1]
v.insert(index, 3 );       // index == 0, v == [3, 4, 8, 5, 1]
index++;                   // index == 1, v == [3, 4, 8, 5, 1]
cout<<v[index]<<endl;      // output: 4
index++;                   // index == 2, v == [3, 4, 8, 5, 1]
index++;                   // index == 3, v == [3, 4, 8, 5, 1]
index++;                   // index == 4, v == [3, 4, 8, 5, 1]
cout<<v[index]<<endl;      // output: 1
v.insert(index, 9 );       // index == 4, v == [3, 4, 8, 5, 9, 1]
index++;                   // index == 5, v == [3, 4, 8, 5, 9, 1]
cout<<v[index]<<endl;      // output: 1
index++;                   // index == 6, v == [3, 4, 8, 5, 9, 1]
v.insert(index, 2 );       // index == 6, v == [3, 4, 8, 5, 9, 1, 2]
cout<<v[index]<<endl;      // output: 2
```

Com veieu a l'exemple d'entrada anterior, hi han espais en blanc envoltant cada número per a facilitar la lectura de l'entrada. Podeu llegir i tractar les comandes així:

```
...
int main()
{
...
string command;
while (cin >> command) {
```

```
if (command == "index++;") {
...
} else if (command == "index--;") {
...
} else if (command == "v.push_front(") {
int number;
cin >> number;
string ending;
cin >> ending; // Això consumeix el ");"
...
} else if (command == "v.push_back(") {
...
} else if (command == "v.insert(index,") {
int number;
cin >> number;
string ending;
cin >> ending; // Això consumeix el ");"
...
} else if (command == "cout<<v[index]<<endl;") {
...
}
}
}
```

Se suposa que la seqüència de comandes és correcta: la variable `index` sempre pren valors entre 0 i la mida actual de `v`, i a més a més, sempre que hi ha una comanda `cout<<v[index]<<endl;`, la variable `index` està entre 0 i la mida actual de `v` menys 1.
Us recomanem que comenceu implementant una solució senzilla que superi els jocs de proves públics, obtenint així la meitat de la nota, i que mireu d'optimitzar-la més tard, si teniu temps.
Podeu utilitzar qualsevol de les estructures de dades presentades al curs (`vector, stack, queue, list, set, map`), i de la forma que considereu oportuna. Fixeu-vos, però, que enfocaments diferents donaran lloc a programes que seran més eficients o menys eficients, i d'això dependrà que pogueu superar només els jocs de proves públics o tots els jocs de proves, cosa que afectarà a la nota.

Entrada

L'entrada del programa és una seqüència de comandes que se suposa que s'executen sobre una variable `index` inicialment a 0, i una "llista" `v` inicialment amb 0 elements. Cada comanda pot ser d'un dels següents tipus:

```
index++;
index--;
v.push_front( NUMBER );
v.push_back( NUMBER );
cout<<v[index]<<endl;
v.insert(index, NUMBER );
```

A on `NUMBER` és un enter qualsevol.

Es garantitza que l'entrada és correcta: la variable `index` sempre pren valors entre 0 i la mida actual de `v`, i a més a més, sempre que hi ha una comanda `cout<<v[index]<<endl;`, la variable `index` està entre 0 i la mida actual de `v` menys 1.

Sortida

Per a cada instrucció `cout<<v[index]<<endl;` el programa escriurà el que suposadament conté la llista a la posició indexada per `index` en aquell moment.

Exemple d'entrada 1

```
v.push_back( 5 );
cout<<v[index]<<endl;
v.push_front( 8 );
cout<<v[index]<<endl;
index++;
cout<<v[index]<<endl;
v.push_back( 1 );
cout<<v[index]<<endl;
index--;
cout<<v[index]<<endl;
v.insert(index, 4 );
v.insert(index, 3 );
index++;
cout<<v[index]<<endl;
index++;
index++;
index++;
cout<<v[index]<<endl;
v.insert(index, 9 );
index++;
cout<<v[index]<<endl;
index++;
v.insert(index, 2 );
cout<<v[index]<<endl;
```

Exemple de sortida 1

```
5
8
5
5
8
4
1
1
2
```

Exemple d'entrada 2

```
v.push_back( -6 );
v.push_back( -2 );
v.insert(index, -1 );
cout<<v[index]<<endl;
cout<<v[index]<<endl;
v.insert(index, 9 );
v.insert(index, -6 );
v.push_front( -3 );
v.insert(index, 6 );
index++;
index++;
cout<<v[index]<<endl;
v.insert(index, -5 );
v.insert(index, 3 );
v.push_back( 9 );
v.insert(index, -7 );
v.insert(index, 7 );
v.push_front( -9 );
v.push_back( 6 );
cout<<v[index]<<endl;
cout<<v[index]<<endl;
```

```
v.push_back( 10 );
v.insert(index, -8 );
v.push_back( 5 );
v.insert(index, -3 );
index--;
index++;
cout<<v[index]<<endl;
v.insert(index, 1 );
index++;
cout<<v[index]<<endl;
cout<<v[index]<<endl;
index++;
index++;
v.insert(index, 8 );
cout<<v[index]<<endl;
cout<<v[index]<<endl;
index++;
v.insert(index, -7 );
v.insert(index, -1 );
v.insert(index, 4 );
v.push_back( 4 );
index--;
index++;
```

```
v.insert(index, 1 );
index++;
v.push_back( -2 );
index--;
index++;
index--;
index++;
v.push_front( -9 );
v.insert(index, 8 );
v.push_front( 8 );
v.insert(index, -3 );
v.push_back( 2 );
v.insert(index, -7 );
index--;
index++;
index++;
index--;
index++;
cout<<v[index]<<endl;
cout<<v[index]<<endl;
cout<<v[index]<<endl;
index++;
index++;
v.push_back( -6 );
v.push_back( -7 );
cout<<v[index]<<endl;
v.push_front( 8 );
cout<<v[index]<<endl;
cout<<v[index]<<endl;
v.insert(index, 4 );
v.insert(index, -2 );
v.push_front( -5 );
v.push_back( 1 );
v.push_front( 1 );
index--;
index++;
v.push_back( 0 );
index++;
v.insert(index, 9 );
v.insert(index, 2 );
cout<<v[index]<<endl;
v.push_front( 1 );
index--;
index++;
cout<<v[index]<<endl;
v.insert(index, -3 );
index--;
index++;
index--;
index++;
v.insert(index, 7 );
index++;
v.push_back( 7 );
v.push_front( -3 );
v.push_front( 4 );
v.push_front( -8 );
cout<<v[index]<<endl;
v.insert(index, 6 );
cout<<v[index]<<endl;
cout<<v[index]<<endl;
v.push_front( 6 );
index++;
v.push_back( 8 );
index--;
index++;
v.push_back( 6 );
v.insert(index, 4 );
```

Exemple de sortida 2

```
-1
-1
-6
-3
-3
-3
-3
-3
8
8
-3
-3
-3
8
8
8
2
-7
-3
6
6
```

Exemple d'entrada 3

```
v.push_back( 1 );
index++;
v.push_back( 2 );
index++;
v.push_back( 3 );
index++;
v.push_back( 4 );
index++;
v.push_back( 5 );
cout<<v[index]<<endl;
index--;
cout<<v[index]<<endl;
index--;
cout<<v[index]<<endl;
index--;
cout<<v[index]<<endl;
index--;
cout<<v[index]<<endl;
```

Exemple de sortida 3

```
5
4
3
2
1
```

Observació

Avaluació sobre 10 punts:

- Solució lenta: 5 punts.
- solució ràpida: 10 punts.

Entenem com a solució ràpida una que és correcta, de cost lineal i capaç de superar els jocs de proves públics i privats. Entenem com a solució lenta una que no és ràpida, però és correcta i capaç de superar els jocs de proves públics.

Informació del problema

Autor : PRO2
Generació : 2023-10-25 19:12:50