**Розробка спеціалізованої мови програмування для робота-маніпулятора у промисловому виробництві**

**Постановка задачі**

Розробка мови для робота-маніпулятора є актуальним завданням, вирішення якого дасть змогу неперервно контролювати весь процес сортування і упакування продукції, забезпечувати ефективність використання робочого часу оператора та забезпечувати безпеку роботи оператора на вищому рівні протягом всього робочого дня.

Процедура сортування і упакування продукції в пакувальну тару існує практично на кожному виробничому підприємстві незалежно від обсягу виробництва. Сортування товару по типу, кольору та іншими ознаками, подальша укладання його в різну тару, з огляду на ті ж ознаки, вибір обгорткового матеріалу прокладки матеріалів тощо. При цьому технологію упаковки / укладання необхідно завжди дотримуватися, тому що від якості упаковки безпосередньо залежить цілісність вмісту тари. До того ж, якщо в сортуванні були допущені помилки, то повернення і заміна товару призведе до відчутної втрати часу і збоїв в логістичних процесах виробника і, можливо, покупця. Виходячи з вище сказаного, вкрай важливо підійти до цього завдання з усією відповідальністю.

Процес сортування і упакування за допомогою роботів-маніпуляторів зображено на (рис. 1). Продукція(праски, телевізори, мультиварки і т.д.) виїжджатимуть з конвеєра по черзі. На ньому ж є датчик проходження продукції, який не дозволить скупчення товарів на одному місці. Поки робот не забере товар з конвеєра, інші будуть залишатися на місці і продовжать рух після того як робот забере вже продукцію. Після того робот буде рухатись до місця упакування товарів в коробки(з урахуванням сортування за приналежністю), а потім попрямує до вантажівки, щоб завантажити товар. Далі знову рухатиметься до конвеєра і продовжить роботу за планом.

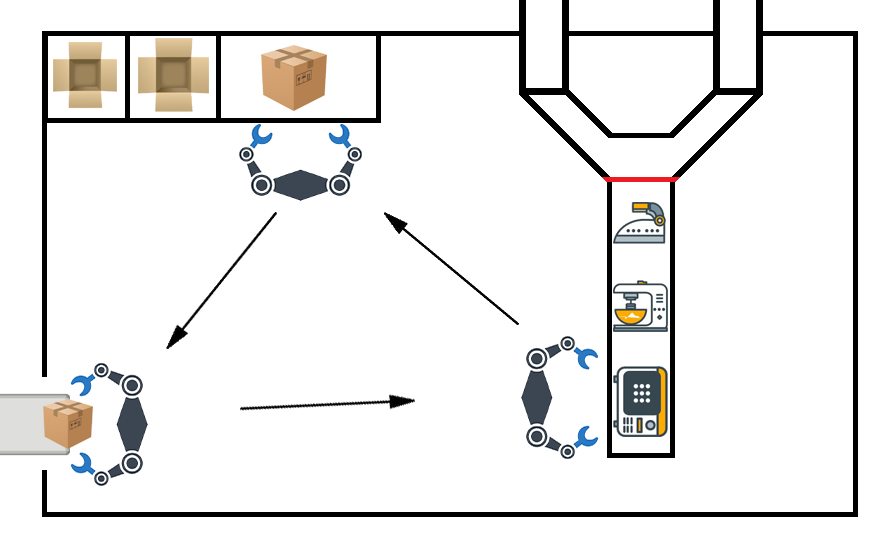
****

Рис. 1- Схематичне зображення дій робота-маніпулятора

Вихідна мова для робота-маніпулятора С#.

**Алфавіт мови**

Алфавіт в собі має такі символи, а саме:

* «А – Я» , «а – я», «A-Z», «a-z»
* «0-9»
* \_ (символ підкреслювання)
* «, . ; : ? ! | / \ ~ \* + -) ( } { < > [ ] % = “»
* пробіл, табуляція, повернення каретки, нова сторінка, новий рядок.

**Перелік операторів**

Для виконання технологічного процесу передбачено 7 команд

* Команди керування роботом
* **ДвижениеВперед** **На** (N);
* **ДвижениеНазад На** (N);
* **ПоворотЛево На** (N);
* **ПоворотПраво На** (N);
* **Взять**;
* **Положить**;
* **Запечатать**;

**Правила синтаксису**

<Программа> ::= ("Script" <Идентификатор>) {<Операция>}

<Идентификатор> ::= (<Буква> | "\_") {(<Буква> | <Цифра> | "\_")}

<Цифра> ::= "0"..."9"

<Буква> ::= <НижнийРегистр> | <ВерхнийРегистр>

<НижнийРегистр> ::= "a"..."z" | "а"..."я"

<ВерхнийРегистр> ::= "A"..."Z" | "А"..."Я"

<Операция> ::= "{" (

<ОперацияИнициализации>

| <ОперацияПрисвоения>

| <ОперацияВыбора>

| <ОперацияЦыкла>

| <ОперацияДвижениеВперед>

| <ОперацияДвижениеНазад>

| <ОперацияПоворотЛево>

| <ОперацияПоворотПраво>

| <ОперацияВзять>

| <ОперацияПоложить>

| <ОперацияЗапечатать>

) "}"

<ОперацияИнициализации> ::= <Идентификатор> ";"

<ОперацияПрисвоения> ::= <Идентификатор> "=" (<Идентификатор> | <Данные>) ";"

<ОперацияВыбора> ::= "ЕСЛИ" "(" (<БулевоВыражение> | <БулевоЗначение>) ")" ":" {<Операция>}

[{"ТОГДА" "(" (<БулевоВыражение> | <БулевоЗначение>) ")" ":" {<Операция>}}]

["ИНАЧЕ" ":" {<Операция>}]

<ОперацияЦыкла> ::= "ВЫПОЛНИТЬ" (<Идентификатор> | <Данные>) "РАЗ" ":" {<Операция>}

<ОперацияДвижениеВперед> ::= "ДвижениеВперед" "На" <ЧисловыеДанные> ";"

<ОперацияДвижениеНазад> ::= "ДвижениеНазад" "На" <ЧисловыеДанные> ";"

<ОперацияПоворотЛево> ::= "ПоворотЛево" "На" <ЧисловыеДанные> ";"

<ОперацияПоворотПраво> ::= "ПоворотПраво" "На" <ЧисловыеДанные> ";"

<ОперацияВзять> ::= "Взять" <ЧисловыеДанные> ";"

<ОперацияПоложить> ::= "Положить" "На" <ЧисловыеДанные> ";"

<ОперацияЗапечатать> ::= "Запечатать" "В" <ЧисловыеДанные> ";"

<Данные> ::= <Буква>

| <Строка>

| <Цифра>

| <Число>

| <ЧислоL>

| <ЧислоСТочкой>

<Строка> ::= '"' {(<Буква> | <Символ> | <СимволЭкранирования> | <Цифра>)} '"'

<Символ> ::= <любой исходный символ, кроме «\» или новой строки>

<СимволЭкранирования> ::= "\" <любой символ ASCII>

<Число> ::= [- | +] {<Цифра>}

<ЧислоL> ::= <Число> ("l" | "L")

<ЧислоСТочкой> ::= <ФормаСТочкой> | <ФормаСЭкспонентой>

<ФормаСТочкой> ::= (<Число> | <ЧислоL>) "." (<Число> | <ЧислоL>)

<ФормаСЭкспонентой> ::= ( (<Число> | <ЧислоL>) | <ФормаСТочкой>) <Экспонентa>

<Экспонентa> ::= ("e" | "E") <Число>

<ЧисловыеДанные> ::= <Цифра> | Число | <ЧислоL> | <ЧислоСТочкой>

<БулевоЗначение> ::= ("true" | "false") | ( "1" | "0" )

<БулевоВыражение> ::= (<Данные> | <Идентификатор> | <АрифметическоеВыражение> | <БулевоВыражение> | <БулевоЗначение>)

<ЛогическийОператор>

(<Данные> | <Идентификатор> | <АрифметическоеВыражение> | <БулевоВыражение> | <БулевоЗначение>)

<АрифметическоеВыражение> ::= (<Идентификатор> | <ЧисловыеДанные>)

<АрифметическийОператор>

(<Идентификатор> | <ЧисловыеДанные> | <АрифметическоеВыражение>)

<ЛогическийОператор> ::= "РАВНО" | "МЕНЬШЕ"

| "БОЛЬШЕ"

| "МЕНЬШЕилиРАВНО" | "БОЛЬШЕилиРАВНО"

| "НЕРАВНО"

<АрифметическийОператор> ::= "+" |"-"

| "\*" | "/"

| "%"

Приклад коду на даній мові:

*Script FulScript*

*{*

*ДвижениеВперед На(34)*

*Взять*

*ДвижениеНазад На(34)*

*ПоворотЛево На(1)*

*ДвижениеВперед На(35)*

*Запечатать*

*ДвижениеНазад На(35)*

*ПоворотЛево На(1)*

*ДвижениеВперед На(15)*

*Положить*

*ДвижениеНазад На(15)*

*ПоворотПраво На(2)*

*}***Правила семантики**

* Кількість роботів – 1;
* Кількість стелажів – 1;
* Кількість полиць на кожному стелажу – 3;
* Робот може повернутися більше ніж на 360 градусів;
* Робоча область 100х100, за яку він не може виїхати ;
* Робот не може взяти деталь з пустої полиці;
* Робот не може покласти деталь на заняту полицю;

**Таблиця конвертації**

|  |  |
| --- | --- |
| Оператор | Конвертація на С#: |
| **ДвижениВперед** **На** N; | goFwd(int N) |
| **ДвижениеНазад На** N; | goBwd(int N) |
| **Поворот** **лево/право На** N; | TurnLeft(TurnSizes N)/ TurnRight(TurnSizes N) |
| **Взять** **;** | Take() |
| **Положить;** | Put(int S) |
| **Запечатать;** | Seal(Objects N) |
| **ЕСЛИ ():**  **{**  **(действие)**  **}**  **ТОГДА():**  **{**  **(действие)**  **}**  **ИНАЧЕ:**  **{**  **(действие)**  **}** | if ()  {  }  else if ()  {  }  else  {  } |
| **ВИКОНАТИ** N **РАЗІВ:**  **{**  **(действие)**  **}** | for(int I = 0; I < N; i++)  {  } |
| **"РАВНО"** | = = |
| **"НЕРАВНО"** | != |
| **"БОЛЬШЕ"** | > |
| **"МЕНЬШЕ"** | < |
| **"МЕНЬШЕилиРАВНО"** | <= |
| **"БОЛЬШЕилиРАВНО"** | >= |

**Вимоги до візуалізації**

1. Анімація з частотою не менше 20 кадрів в секунду.
2. Анімація пересування.
3. Анімація захоплення предмета.
4. Анімація переміщення предмета.
5. Покажчик напрямку робота.
6. Показник наявності предмета у робота.
7. Контрастні кольори об'єктів.

**Вимоги до переривань**

* 1. Переривання при виявленні сторонніх об'єктів на території (робот припинить рух, і відновить при відсутності стороннього об'єкта).
  2. Переривання при аварійній ситуації (пожежа/землетрус).
  3. Переривання оператора.

**Вимоги до Log-файлу**

Програма під час своєї роботи повинна зберігати всі дані про роботу в лог-файлі. Чим краще лог-файл, тим простіше знаходити та вирішувати проблеми. Виділимо декілька основних концепцій для зберігання інформації в таких файлах:

1. Лог повинен відображати всі важливі події. Наприклад: запуск програми, її закінчення, виникнення переривань, виникнення помилок і так далі.
2. Їх повинно бути легко фільтрувати. Необхідно подбати про унікальність тих слів / символів, за якими можливий пошук, і виключити часто повторювані слова.
3. Додавання [TAGS] для маркування концепцій програми. Наприклад: [INTERRUPT], [ROBOT], [ERROR] і так далі. Якщо шукати за цими тегами, можна отримати зріз логів, які зазвичай розкидані по декількох класах.
4. Один рядок - одне повідомлення. Варто обмежитися виведенням одного повідомлення на рядок, інакше пошук буде не такий ефективний.
5. Логів не повинно бути занадто багато і вони не повинні бути надмірними - зайва інформація буде засмічувати екран. Тільки саме важливе і потрібне.
6. Формат файлу повинен бути розповсюдженим і не потребувати стороннього ПЗ для його перегляду ч редагування.

Тому, лог-файл буде створений за наступними вимогами:

1. Формат файлу - .txt
2. Форма запису – [Дата/Час][TAG]-Повідомлення логу