**Автоматизация и моделирование в проектировании и управлении. 2023. № 2 (20). С. 4-13.**
**ISSN 2658-3488 print, ISSN 2658-6436 online**
**Automation and modeling in design and management. 2023. № 2 (20). P. 4-13.**

## *Автоматизация и управление технологическими процессами и производствами, системы автоматизации проектирования*

Научная статья
Статья в открытом доступе
УДК 004.4.242
DOI 10.30987/2658-6436-2023-2-4-13

# ПРИМЕНЕНИЕ ТЕХНОЛОГИИ АВТОМАТНОГО ПРОГРАММИРОВАНИЯ ДЛЯ РАЗРАБОТКИ ПРОГРАММЫ УПРАВЛЕНИЯ СКЛАДСКИМИ ПОМЕЩЕНИЯМИ НА ПРЕДПРИЯТИЯХ

**Павел Васильевич Гумунюк[1]✉, Александр Николаевич Шурпо[2]✉**
[1, 2] Институт конструкторско-технологической информатики Российской академии наук, г. Москва, Россия

[1] pavel05091997@yandex.ru, http://orcid.org/0000-0000-0000-0000
[2] a-shurpo@yandex.ru, http://orcid.org/0000-0003-1962-1969

**Аннотация.** *В статье наглядно показано, что разработка системы управления автоматических платформ для складского помещения можно реализовать на основании принципов и моделей автоматного программирования, при условии применения принципов работы системы и выделения устойчивых состояний, в которых эта система может находиться. Цель данной статьи, помимо наглядного применения принципов и методов автоматного программирования для написания системы управления и демонстрации возможностей Switch-технологии, является её популяризация и распространение информации о ней, а также о возможностях её применения в широком спектре программирования различных умных устройств, промышленных установок и программ, функционирующих в различных отраслях. Автоматное программирование подразумевает подробное изучение технологического процесса с целью явного выделения устойчивых состояний исследуемой системы, которые в дальнейшем применяются для построения логики автоматизируемой установки – этому и посвящена первая часть статьи. Следующий шаг – написание самой логики, основанной на условиях переключения системы из одного устойчивого состояния к следующему. Выбранная автоматизированная система обладает определенной сложностью из-за возможности задействования большого количества объектов при её построении, однако изменение количества объектов в системе, с целью расширения или минимизации производства, не требует полной замены управляющей программы в виду гибкости её логики, написанной методом автоматного программирования.*

**Ключевые слова:** автоматическая передвижная платформа, автоматическое складское помещение, умное хранилище товаров, Switch-технология, автомат, автоматное программирование, система управления, индустрия 4.0

**Для цитирования:** Гумунюк П.В., Шурпо А.Н. Применение технологии автоматного программирования для разработки программы управления складскими помещениями на предприятиях // Автоматизация и моделирование в проектировании и управлении. 2023. №2 (20). С. 4-13. doi: 10.30987/2658-6436-2023-2-4-13.

Original article
Open Access Article

# APPLYING AUTOMATIC PROGRAMMING TECHNOLOGY FOR DEVELOPING A WAREHOUSE MANAGEMENT PROGRAMME AT ENTERPRISES

**Pavel V. Gumunyuk[1]✉, Alexander N. Shurpo[2]✉**

[1, 2] Institute for Design-Technological Informatics of the Russian Academy of Sciences, Moscow, Russia

[1] pavel05091997@yandex.ru, http://orcid.org/0000-0000-0000-0000
[2] a-shurpo@yandex.ru, http://orcid.org/0000-0003-1962-1969

**Abstract.** *The article clearly shows that developing a control system for automatic storage platforms can be implemented on the basis of the principles and models of automatic programming, subject to applying the system principles and identifying stable states in which this system can be. The aim of the article, in addition to the visual application of automatic programming principles and methods for writing a control system and demonstrating the Switch technol-*

*ogy capabilities, is its promotion and dissemination, as well as the possibilities of its application in a wide range of programming various smart devices, industrial installations and programmes operating in various industries. Automatic programming implies a detailed study of the technological process to explicitly identify the stable states of the system under study, which are subsequently used to build automated installation logic; this is the subject of the first part of the article. The next step is to write the logic itself, based on the conditions for switching the system from one stable state to the next. The selected automated system has a certain complexity due to the possibility of involving a large number of objects in its construction, however, changing the number of objects in the system to expand or minimize production, does not require a complete replacement of the control programme due to the flexibility of its logic written by automatic programming.*

***Keywords:*** automatic mobile platform, automatic storage room, smart storage of goods, Switch technology, machine, automatic programming, control system, industry 4.0

**For citation:** Gumunyuk P.V., Shurpo A.N. Applying automatic programming technology for developing a warehouse management programme at enterprises. Automation and modeling in design and management, 2023, no. 2 (20). pp. 4-13. doi: 10.30987/2658-6436-2023-2-4-13.

## Введение

Цифровизация всё более активно проникает в нашу современную жизнь, поэтому целесообразно рассматривать технологии, которые можно отнести к умным предприятиям будущего. Такие производства активно применяют автоматизированные платформы для перемещения заготовок по территории завода между цехами.

## Материалы, модели, эксперименты и методы

Рассмотрим процесс программирования такой платформы путём выделения явных состояний и использования Switch-технологии. Эта технология проста в освоении и не требует от разработчика программного кода слишком глубоких знаний программирования, однако для практичного применения этой технологии, необходимо обязательно знать устройство системы управления. Далее показано, как знание устройства системы управления и явное выделение состояний позволяет сформировать графы переходов, по которым можно сгенерировать программный код системы управления. В тексте книги «Преобразование графов переходов, представленных в формате MSVisio, в исходные коды программ для различных языков программирования (инструментальное средство MetaAuto)» рассматриваются основные логические паттерны автоматного программирования в виде компьютерной модели как словаря понятий [1].

Switch-технология является еще одним названием технологии автоматного программирования, поскольку использует оператор множественного выбора switch для выбора устойчивого состояния, в чём можно убедиться, наблюдая приведённый ниже код. Для программирования платформы, необходимо выяснить технические средства, которыми она оснащена.

Видеодатчик считывает линию с пола цеха, сопоставляя её с маршрутом, указанным в процессоре платформы; сервоприводы приводят платформу в движение, а реле отвечает за разворот платформы.

Для рассмотрения программы управления платформой через призму автоматного программирования необходимо выделить её основные состояния. Платформа может, как стоять на месте, так и двигаться. Это два её основных состояния. Если платформа стоит на месте – видеодатчик ничего не считывает, реле находится в прямом положении, сервоприводы выключены. Когда же платформа движется – видеодатчик считывает данные с линий разметки в цеху и сопоставляет их с выбранным маршрутом; реле в зависимости от направления маршрута и положения платформы на этом маршруте находится в одном из трёх положений – прямо, право или влево; сервоприводы запущены.

Автоматическая платформа весьма удобна тем, что является эффективным решением по транспортировке грузов, как на производстве, так и в хранилищах [2]. В случае, когда платформа пуста, то она должна вернуться в начальное состояние, из которого производится выдача команд и груза. Если по завершению технологический операции груз не был передан, то платформа должна остановиться и повторить сигнал о подаче груза.

Заряда аккумулятора платформы хватает на 7 часов непрерывной работы. Если заряд опустится ниже значения в 40 %, то платформа должна завершить начатую операцию, на новую операцию не откликаться и отправиться на зарядку аккумулятора.

Для управления операциями платформ, а также для предотвращения угрозы их столкновения необходим контроллер, отвечающий за координацию их действий. Если на пути платформы возникнет препятствие, то она должна выбрать альтернативный маршрут для передвижения, избегая выбора аварийных маршрутов.

Видеодатчик состоит из трех датчиков, задача которых обнаружение белой линии, по которой должна следовать тележка. Об обнаружении белой линии должен сообщать центральный датчик. Если происходит отклонение от маршрута движения тележки, то, в первую очередь, опрашивается тот датчик, который последним фиксировал белую линию. Если он её не обнаруживает в течение 15 секунд, то происходит опрос датчика с противоположенной стороны и если этот датчик в течение 15 секунд также не дает ответа, то тележке необходимо совершить поворот на 180 градусов по часовой стрелке пока центральный датчик не зафиксирует белую полосу. По обнаружению полосы тележка должна совершить ещё один оборот по часовой стрелке на 180 градусов для обнаружения белой линии центральным датчиком. Если тележка не нашла белую линию после второго оборота на 180 градусов, то ей необходимо развернуться и, обнаружив белую линию, вернуться назад к началу своего маршрута, в то время как система должна подобрать для тележки альтернативный маршрут. Если тележка не смогла найти и первую линию, то ей необходимо совершить ещё оборот на 180 градусов и попытаться найти первоначально требуемую линию ещё раз, если это не произошло, то тогда тележка должна остановиться на месте и сообщить об останове технологического участка и об вызове оператора для устранения неисправности. Если на каком-либо технологическом участке происходит потеря тележками маршрута два раза подряд, то данный участок обозначается как неисправный и исключается из вариантов маршрута, пока оператор не снимет этот флаг вручную [3].

На основании рассмотренной системы управления можно выделить несколько автоматов:

– Класс Visualizator отвечает за визуализации технологического процесса, а также снабжение интерфейсом для управления передвижениями платформ;

– Класс MainCore является ядром системы и служит в качестве управляющего контроллера;

– Класс AutoPlatform управляет поведением автоматизированной тележки и является главным рабочим звеном системы;

– Класс Store будет обозначать маршруты, по которым могут двигаться тележки и передавать в класс MainCore информацию о загруженности маршрутов.

Простейшая схема складского помещения, состоящая из 3-х автоматических тележек и 9-ти стеллажей с дорожками между ними представлена на рис. 1 (A, B, C – автоматические тележки на своих начальных позициях; 1 – 9 – стеллажи; I – IX – узлы в дорожках, между которыми образуются 12 секторов).

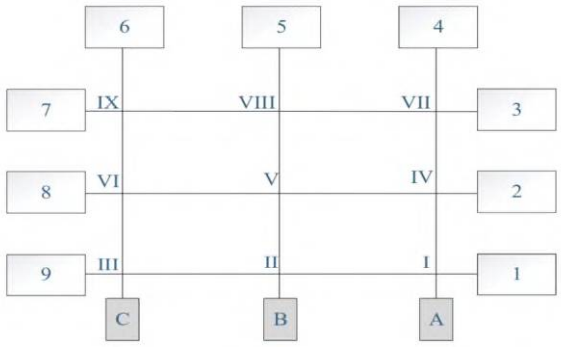



**Рис. 1. Схематическое представление складского помещения**
***Fig. 1. Schematic representation of the warehouse***

Если путь не занят и путь не остановлен, то выбрать нужно самый правый путь из двух предлагаемых вариантов. Второй путь пусть остается свободным для варианта, если на первом возникнет неисправность.

Класс управления маршрутами является единственным, который может быть заменен на такой же класс, но с другими параметрами, так как не существует единого решения относительно количества помещений, в которые будет доставляться груз и количества маршру-

тов, выделенных для того или иного помещения. Для решения данной задачи взят склад, состоящий из трех помещений и четырех маршрутов в каждое помещение; общее количество платформ будет равным шести. Платформа должна двигаться по правому пути, а если правый путь будет занят, то платформа должна переместиться на левый путь. Если оба пути заняты, то платформа должна выбрать свободный путь другой платформы. Если же все возможные пути заняты, то платформа должна ждать пока, какой-либо из возможных путей, не освободится. Чтобы знать, какой путь свободен, а какой нет, дорожки, разбитые на участки, должны сообщать о нахождении на них тележек, а если платформа застряла на границе двух участков, то оба участка должны сигнализировать о том, что она находится на них [4].

Методика составления графов перехода и генерации кода может применяться для составления классов с измененными параметрами.

Целесообразно разделить программу на три составные части:

– программа управления автоматической платформой;

– программа управления контроллером автоматических платформ;

– программа управления складским помещением.

Логика программы управления автоматической платформой в виде блок-схемы представлена на рис. 2.

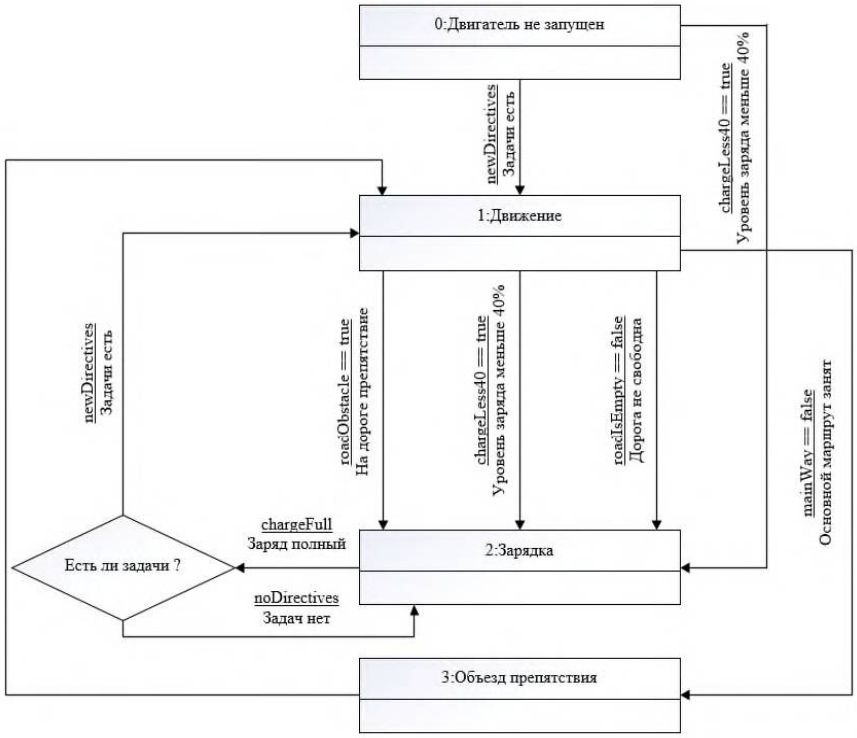



**Рис. 2. Представление программы управления автоматической платформы в виде алгоритма (блок-схемы)**

***Fig. 2. Representation of the automatic platform control program code in the form of an algorithm (block-diagram)***

В программе управления автоматической платформой задействованы следующие переменные, каждая из которых отображает параметры автоматизированной системы:

int relay; // реле управления может принимать следующие значения: 0 – поворот налево; 1 – езда прямо; 2 – поворот направо.

int videoSensor; // видеодатчик принимает четыре основных значения: 0 – линия слева; 1 – линия посередине; 2 – линия справа; 3 – линия утеряна, сравнивает имеющийся маршрут с полученным из контроллера значением.

int lineSensor1; // датчик обнаружения белой линии левый.

int lineSensor2; // датчик обнаружения белой линии средний.

int lineSensor3; // датчик обнаружения белой линии правый.

int platformState; //

bool cargoReady; // груз загружен.

bool cargoEmpty; // груз снят.

bool engineStart; // двигатель запущен.

```
bool turnRelay; // реле поворота включено.
bool chargeMore40; // заряд выше 40 %.
bool chargeLess40; // заряд ниже 40 %.
bool roadIsEmpty; // дорога свободна.
bool roadObstacle; // на дороге препятствие.
bool alternativeWay; // альтернативный маршрут.
bool mainWay; // основной маршрут.
bool noDirectives; // задач нет.
bool newDirectives; // задачи есть.
bool chargerSwitchOn; // зарядка подключена.
bool chargerSwitchOff; // зарядка отключена.
bool chargeFull; // заряд полный немаловажных.
switch ( platformState )
{
 case 0: // двигатель не запущен.
 cargoReady = false;
 cargoEmpty = false;
 engineStart = false;
 turnRelay = false;
 chargeMore40;
 if ( chargeLess40 ) {
  platformState ==2; }
if (newDirectives == true ) {
  platformState ==1; }
 roadIsEmpty = false;
 roadObstacle = false;
 alternativeWay = false;
 mainWay = false;
 noDirectives = true;
 chargerSwitchOn = false;
 chargerSwitchOff = false;
 chargeFull = false;
 break;
 case 1: // движение.
 cargoReady = true;
 cargoEmpty = false;
 engineStart = true;
 turnRelay = true;
 chargeMore40;
 if ( chargeLess40 == true) {
  chargerSwitchOn = true;
  chargerSwitchOff = false;
  chargeFull = false;
  platformState ==2; }
 if (roadIsEmpty == false) {
  chargerSwitchOn = true;
  chargerSwitchOff = false;
  chargeFull = false;
  platformState ==2; }
 if ( roadObstacle == true ) {
  chargerSwitchOn = true;
  chargerSwitchOff = false;
  chargeFull = false;
  platformState ==2; }
 if ( mainWay == false ) {
  alternativeWay == true;
```

```
 platformState ==3; }
noDirectives = false;
newDirectives = true;
break;
case 2: // зарядка.
if ( chargeFull == true ) {
 if ( newDirectives == true ) {
  platformState == 1; }
 else if ( noDirectives == true ) {
 platformState == 2; }
}
break;
case 3: // объезд препятствия.
alternativeWay = true;
platformState ==1;
break; }
```

Программа управления контроллером автоматических платформ отвечает за распределение обязанностей автоматических платформ и за загруженность линий при их движении по маршруту [6].

Путь к каждому стеллажу состоит из определенного количества секторов, причём каждый сектор отделен от предыдущего узлом. Для схемы, применяемой в этом примере, получим 9 узлов и 12 секторов. Если сектор между узлами неисправен или на нём имеется неисправная тележка, тогда узлы (до него и после него) сигнализируют об этом и перенаправляют другие тележки в обход налево (совершая проезд через три сектора и четыре узла), выполняя при этом шесть действий, а именно: поворот налево; проезд вперед; поворот направо; проезд вперед; поворот направо; проезд вперед.

Для определения занятости маршрута нужно сигнализировать о наличии тележки или неисправности на маршруте – всего мы имеем 12 сегментов.

Пусть путь тележки 3 лежит на стеллаж 1. Тележке необходимо пройти оптимальным путём – это сегмент 1 и сегмент 2. Сегмент 2 сигнализирует о поломке, следовательно, его надо объехать. Путь лежит через сегменты налево, направо, направо, вернуться на маршрут.

Выделим состояния. Начальное состояние. Все тележки на своих местах; каждой тележке ищется путь; тележке ищется обходной путь и устранение неполадок, связанных с тележками [5].

Программа управления контроллером автоматических платформ назначает задачу первой по порядку свободной тележке. Тележке, имеющей задачу, присваивается статус занятой, она освобождается от новых задач, пока не окажется свободной, тогда новую задачу выполняет следующая по порядку тележка. Во избежание износа только одной тележки необходимо ввести счётчик работ, выполненных тележками. Если у тележки 1 работ больше, чем у тележки 2 и при этом обе тележки простаивают, нужно присвоить работу тележке 2. Если у тележки 2 работ больше, чем у тележки 3 и при этом обе тележки простаивают, то присвоить работу тележке 3. Если у всех трёх тележек одинаковое количество работ, то нужно обнулить счётчик и присвоить работы тележке 1. Здесь мы имеем два типа состояний: во-первых, состояния контроллера. Контролер свободен, контролер задал задачу тележке *n*, контроллер не имеет свободных тележек (режим ожидания).

Логика программы управления контроллером автоматических платформ в виде блок-схемы представлена на рис. 3.

Определимся с переменными:

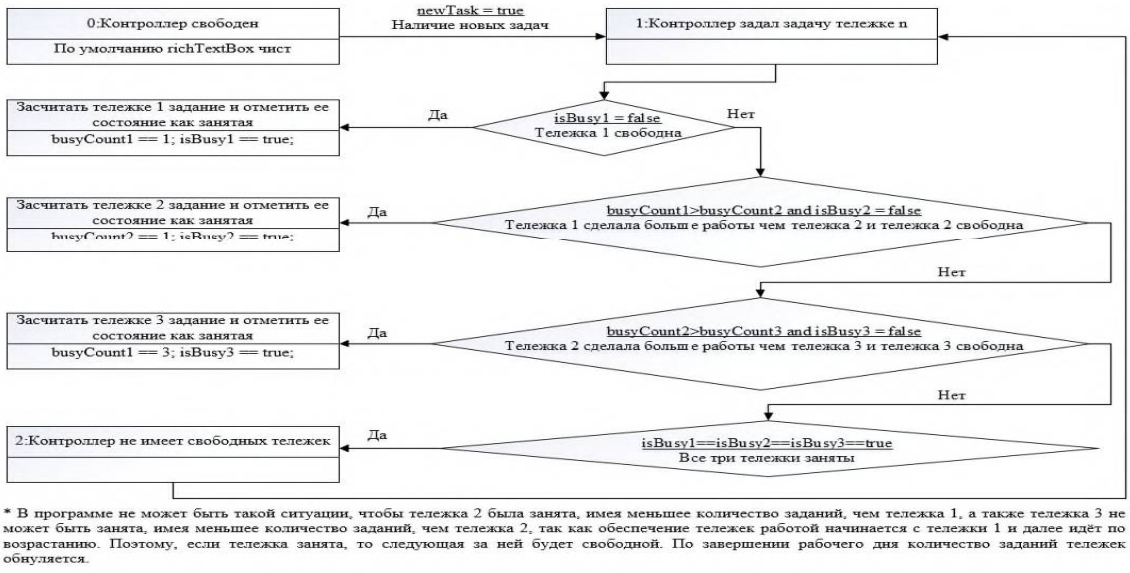
bool newTask; // наличие новых задач.

int PlatformId1,2,..,n; // порядковый номер тележки, в данном примере таких переменных 3, для каждой тележки по одной штуке.

bool isBusy1,2,...,n; // флаг занятости тележки, в данном примере таких переменных 3, для каждой тележки по одной штуке.

int busyCount1,2,...,n; // счетчик работ выполненных тележкой, в данном примере таких переменных 3, для каждой тележки по одной штуке.

```
into controllerBusyness // состояния контроллера.
switch ( controllerBusyness )
{
 case 0: // контроллер свободен.
 if (newTask = true ) {
  controllerBusyness == 1;
  newTask == false; }
 break;
 case 1: // контролер задал задачу тележке n.
 if ( isBusy1 = false ) {
  busyCount1 == 1;
  isBusy1 == true; }
 else if ( busyCount1>busyCount2 and isBusy2 = false ) {
  busyCount2 == 1;
  isBusy2 == true; }
 else if ( busyCount2>busyCount3 and isBusy3 = false ) {
  busyCount3 == 1;
  isBusy3 == true; }
else if ( isBusy1==isBusy2==isBusy3==true ) {
  controllerBusyness == 2; }
 break;
 case 2: // контроллер не имеет свободных тележек (режим ожидания).
 wait 100;
 controllerBusyness == 1;
 break; }
```




* В программе не может быть такой ситуации, чтобы тележка 2 была занята, имея меньшее количество заданий, чем тележка 1, а также тележка 3 не может быть занята, имея меньшее количество заданий, чем тележка 2, так как обеспечение тележек работой начинается с тележки 1 и далее идёт по возрастанию. Поэтому, если тележка занята, то следующая за ней будет свободной. По завершении рабочего дня количество заданий тележек обнуляется.

**Рис. 3. Представление программы управления контроллером автоматических платформ в виде алгоритма (блок- схемы)**

***Fig. 3. Representation of the program for controlling the controller of automatic platforms in the form of an algorithm (block-diagram)***

Для разбора программы управления складским помещением на наглядном примере, возьмём схему из 3-х автоматизированных платформ (тележек) и 9-ти складских хранилищ (стеллажей). На местах пересечения путей образуются узлы, а участки между ними являются секторами. Данная схема является довольно простой, а реальные складские помещения имеют дело с куда большими цифрами. Назначение данной схемы – показать опытным путём пример составления управляющей логики для автоматизированной складской системы, используя явное выделение состояний и применяя технологии автоматного программирования. Данная система управления, может быть, масштабирована для больших складских хозяйств

путём редактирования базы данных маршрутов тележек и поэтапного расширения числа используемых тележек и мест для их хранения. При этом увеличивать площадь путей не обязательно, так как разметка, состоящая из узлов и секторов, не требует для себя больших пространств.

С текстами программ можно ознакомиться в репозитории https://github.com/pavel–proger/SmartStorage.

Состояния могут быть следующими:

– тележка *x* едет к стеллажу y по пути № *t*;

– тележка *x* объезжает препятствие на секторе *a* через узел *b*;

– тележка *x* едет на базу по маршруту № *t*;

– тележка *x* едет к стеллажу *z* по маршруту *c*;

– тележка *x* разгружается или загружается;

– тележка *x* на базе и ждет дальнейших команд.

Все тележки, если они не заняты, обязаны возвращаться на базу для того, чтобы не создавать помехи.

Логика программы управления складским помещением в виде блок-схемы представлена на рис. 4.

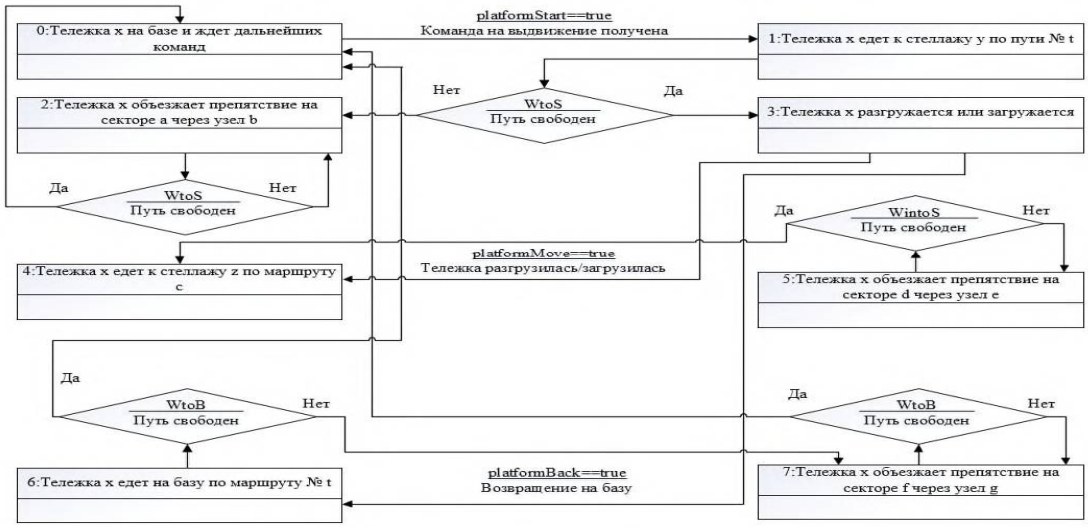



**Рис.4. Представление программы управления складским помещением в виде алгоритма (блок-схемы)**
***Fig.4. Representation of the warehouse management program in the form of an algorithm (block-diagram)***

Определимся с переменными:

```
bool platformStart; // команда на выдвижение получена.
bool WtoS; // сигнализация свободного пути при движении вперед.
bool platformMove; //сигнализация загрузки или разгрузки платформы на стеллаже.
bool platformBack; // команда на возвращение на базу получена.
bool WtoB; // сигнализация свободного пути при движении назад.
bool WintoS; // сигнализация свободного пути при движении по складу.
switch ( platformWay )
{
 case 0: // тележка x на базе и ждет дальнейших команд.
 if ( platformStart == true ) {
  platformWay = 1;
  platformStart = false; }
 break;
 case 1: // тележка x едет к стеллажу y по пути № t.
 if ( WtoS == false) {
  platformWay = 2; }
 else if ( WtoS == true ) {
  platformWay = 3; }
 break;
 case 2: // тележка x объезжает препятствие на секторе a через узел b.
```

```
if ( WtoS == false) {
 platformWay = 2; }
else if ( WtoS == true ) {
 platformWay = 1; }
break;
case 3: // тележка x разгружается или загружается.
if ( platformMove == true ) {
 platformWay = 4; }
else if ( platformBack == true) {
 platformWay = 6; }
break;
case 4: // тележка x едет к стеллажу z по маршруту c.
if ( WintoS == false ) {
 platformWay = 5; }
else if ( WintoS == true ) {
 platformWay = 6; }
break;
case 5: // тележка x объезжает препятствие на секторе d через узел e.
if ( WintoS == false ) {
 platformWay = 5; }
else if ( WintoS == true ) {
 platformWay = 4; }
break;
case 6: // тележка x едет на базу по маршруту № t.
if ( WtoB == false ) {
 platformWay = 7; }
else if ( WtoB == true ) {
 platformWay = 0; }
break;
case 7: // тележка x объезжает препятствие на секторе f через узел g.
if ( WtoB == false ) {
 platformWay = 7; }
else if ( WtoB == true ) {
 platformWay = 0; }
break; }
```

## Результаты

Использование Switch-технологии позволило визуализировать систему распределения задач между автоматизированными платформами для оптимального управления складским помещением [7]. Общий код программы имеет три неотъемлемые части, образующие совместно звенья автоматизированной системы. При этом в каждом звене, в качестве основного логического решения, нашлось место выделенным состояниям, переключение между которыми осуществляется при помощи оператора switch. Данный подход к построению систем управления позволяет подтвердить принцип того, что любая система может рассматриваться как структура, функционирующая при помощи нахождения её в определенных состояниях, а переход между этими состояниями осуществляется благодаря внешним или внутренним событиям системы [8].

## Заключение

В настоящее время в самых различных отраслях всё больше становится умных производств, которые могут быть реализованы как на малых, так и на средних и на больших предприятиях. Все эти предприятия, как правило, имеют, складские помещения, которые, являясь неотъемлемой частью производства, требуют автоматизации данных структур, что позволит улучшить эффективность предприятий в целом [9].

Умное хранилище товаров на предприятиях позиционирует себя как стоящее дополнение автоматизированного производства в рамках индустрии 4.0 [10].

## Список источников:

1. Канжелев С.Ю., Шалыто А.А. Преобразование графов переходов, представленных в формате MSVisio, в исходные коды программ для различных языков программирования (инструментальное средство MetaAuto). – Санкт–Петербург: ИТМО, 2005. – 102 с.
2. Черепанов П.Ю., Романов П.А. Разработка автоматизированной системы для перевозки легких грузов. – Наука, техника и образование. – 2017. – С. 10-16.
3. Шалыто А.А. Методы аппаратной и программной реализации алгоритмов. – Санкт–Петербург: ИТМО, 2005. – 779 с.
4. Normand V., Exertier D. «Model–driven systems engineering: SysML & the MDSysE approach at Thales», in «Model Driven Engineering for distributed real–time embedded systems», John Wiley & Sons, Sept. 2005, 288.
5. INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, 4th Edition, Wiley, 2015, 305.
6. Шалыто А.А., Туккель Н.И. Switch-технология – автоматный подход к созданию программного обеспечения «реактивных» систем. – Программирование. – 2001. – №5. – С. 45-62.
7. Поликарпова Н.И., Шалыто А.А. Автоматное программирование СПбГУ ИТМО. 2008. – 167 с.
8. Татарчевский В.А. Switch-технология в задачах логического управления. – Программные продукты и системы. – №5. – 2006. – С. 30-32.
9. Липкин Е. ИНДУСТРИЯ 4.0: Умные технологии – ключевой элемент в промышленной конкуренции. – М.: ООО «Остек–СМТ», 2017. – 224 с.
10. Шалыто А.А. Алгоритмизация и программирование задач логического управления. – СПбГУ ИТМО, – 1998. – 55 с.

## References:

1. Kanzhelev S.Yu., Shalyto A.A. The Use of Graph Diagrams and Transition Graphs in MSVisio Format Into Program Source Codes for Various Programming Languages (MetaAuto tool). Saint Petersburg: National Research University Institute of Fine Mechanics and Optics; 2005.
2. Cherepanov P.Yu., Romanov P.A. The Development of an Automated System for Transporting Light Loads. Science, Technology and Education. 2017:10-16.
3. Shalyto A.A. Methods of Hardware and Software Algorithms Implementation. Saint Petersburg: National Research University Institute of Fine Mechanics and Optics; 2005.
4. Normand V, Exertier D. Model-Driven Systems Engineering: SysML & the MDSysE Approach at Thales. In: Model Driven Engineering for Distributed, Real-Time and Embedded Systems. New-York: John Wiley & Sons; 2005.
5. INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, 4th Edition. New-York: Wiley, 2015.
6. Shalyto A.A., Tukkel N.I. Switch-Technology: an Automatic Approach to Developing Software for Reactive Systems. Programming and Computer Software. 2001;5:45-62.
7. Polikarpova N.I., Shalyto A.A. Automated-Based Programming. Saint Petersburg: National Research University Institute of Fine Mechanics and Optics; 2008.
8. Tatarchevskii V.A. Switch-Technology in Logical Control Problems. Software and Systems. 2006;5:30-32.
9. Lipkin E. Industry 4.0: Smart Technology is a Key Element in Industrial Competition. Moscow: Ostek-SMT; 2017.
10. Shalyto A.A. Algorithmization and Programming of Problems of Logical Control. Saint Petersburg: National Research University Institute of Fine Mechanics and Optics; 1998.



### Информация об авторах:

**Гумунюк Павел Васильевич** – аспирант института конструкторско-технологической информатики Российской академии наук

**Шурпо Александр Николаевич** – кандидат технических наук, старший научный сотрудник института конструкторско-технологической информатики Российской академии наук, ORCID:0000-0003-1962-1969

### Information about authors:

**Gumunyuk Pavel Vasilievich** – postgraduate student, Institute for Design-Technological Informatics of the Russian Academy of Sciences

**Shurpo Alexander Nikolaevich** – Candidate of Technical Sciences, Senior Researcher, Institute for Design-Technological Informatics of the Russian Academy of Sciences, ORCID:0000-0003-1962-1969



**Вклад авторов: все авторы сделали эквивалентный вклад в подготовку публикации.**
**Contribution of the authors: the authors contributed equally to this article.**

**Авторы заявляют об отсутствии конфликта интересов.**
**The authors declare no conflicts of interests.**

**Статья поступила в редакцию 28.02.2023; одобрена после рецензирования 21.03.2023; принята к публикации 28.03.2023.**
**The article was submitted 28.02.2023; approved after reviewing 21.03.2023; accepted for publication 28.03.2023.**

**Рецензент** – Аверченков А.В., доктор технических наук, профессор, Брянский государственный технический университет.
**Reviewer** – Averchenkov A.V., Doctor of Technical Sciences, Professor, Bryansk State Technical University.