UNIVERSIDAD DE LAS AMÉRICAS PUEBLA

Escuela de Ingenierías

Departamento de Ingeniería Industrial y Mecánica

Desarrollo de alternativas de layout para estacionamientos

Tesis que, para completar los requisitos del Programa de Honores presenta la estudiante

Araceli de Jesús Córdova Chávez

162755

Licenciatura en Ingeniería Industrial

Directora de Tesis: Dra. Carmen Xochitl Flores Mendoza

San Andrés Cholula, Puebla Verano I 2022

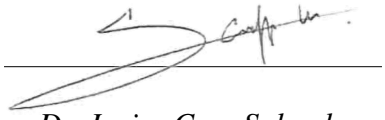
Tesis que, para completar los requisitos del Programa de Honores presenta la estudiante

Araceli de Jesús Córdova Chávez 162755

Director de tesis

Dra. Carmen Xóchitl Flores Mendoza

Presidente de tesis



Dr. Javier Cruz Salgado

Secretario de tesis

Dra. Elsa Velázquez Miranda

Agradecimientos

Quiero agradecer a mis profesores, quienes me impartieron sus conocimientos a lo largo de mi carrera universitaria. En particular, a mi profesora y tutora Carmen Xóchitl Flores Mendoza por su gran apoyo en el desarrollo de mi tesis.

A la UDLAP por otorgarme su educación y recursos para estudiar.

A mi familia, quienes siempre han estado a mi lado brindandome palabras de aliento. Especialmente, a mis padres y hermana, porque son las personas más cercanas a mí.

Índice general

Índice de figuras

Capítulo 1

Resumen

1.1. Introducción

Se sabe de estudios que la captación y/o conservación de clientes de un negocio se encuentra relacionado con la disponibilidad de estacionamiento. Los autores Moreno Rincón et al. (2017) publican que el 60 % de los visitantes encuestados perciben la facilidad de llegada (peatonal o vehicular) en 4, tomando como referencia la escala de medición del 1 al 4, siendo 4 muy importante, para asistir al Cacique centro comercial ubicado en Bucaramanga, Santander, Colombia. También, se indica que el 57 % ingresa en vehículo particular y el 16 % en taxi. El 68 % de los comerciantes encuestados perciben la facilidad de llegada (peatonal o vehicular) en 4 para asistir al centro comercial. A su vez, los comerciantes expresan que la percepción general buena y excelente de ellos mismos sobre la movilidad de desplazamiento en el centro comercial se debe a las vías de acceso, amplitud de estacionamiento y organización.

La tesis presentada se realiza con el fin de ofrecer una alternativa para desarrollar propuestas de layout para un estacionamiento. Para ello, se divide en cuatro capítulos: resumen, marco teórico, metodología y conclusiones y recomendaciones. En el primer capítulo se le da sentido a la tesis, es decir, se presenta la justificación, objetivo, alcance, limitaciones y descripción general de la metodología empleada para el desarrollo de alternativas de layout. El segundo capítulo explica el contexto que debe ser tomado en cuenta para el desarrollo de alternativas de layout. El tercer capítulo, explica cómo se desarrolla la propuesta de layout para un estacionamiento descrito por los autores Tompkins et al. (2010) y se implementa empleando programación con el lenguaje Python.

1.2. Justificación

Litman (2016) señala que 23 horas al día se encuentra estacionado un automóvil típico y que los automóviles deben de ser estacionados en cada destino. También, que los estacionamientos son de los problemas más comunes que enfrentan los diseñadores, operadores y

aquellos en términos de gestión. Como lo indica Litman (2016), una buena gestión trae como beneficios ahorrar costos del estacionamiento, mejorar la calidad del servicio, abordar los requerimientos del estacionamiento, entre otros. Al interpretar las palabras del autor, se cree que los estacionamientos merecen atención ya que son ampliamente usados y comúnmente problemáticos al mismo tiempo. Por esta razón, se decide programar un algoritmo en Python que permita desarrollar el layout de un estacionamiento.

1.3. Objetivo

Desarrollar alternativas de layout para un estacionamiento, aplicando la metolodogía de Tompkins et al. (2010) y programación con el software Python, con el propósito de proporcionar una herramienta que facilite desarrollar el layout al público en general.

1.4. Alcance

El presente estudio considera medidas de ancho y ángulo para el cajón de estacionamiento, tipo de módulo para el estacionamiento y clasificación de los automóviles empleados por expertos como Ramsey y Sleeper (2011) del área de arquitectura, y referidas también por los expertos Tompkins et al. (2010) del área de ingeniería industrial. El presente estudio termina cuando se elabora el trazo de la propuesta de layout del estacionamiento con base a los resultados obtenidos del programa en Python.

1.5. Limitaciones

No se incluye un estudio de mercado con el cuál se determine la demanda de cajones por estacionamiento. Este dato es proporcionado por el usuario. Por tanto, el programa propuesto en Python lo considera como dato de entrada.

1.6. Metodología

La metodología empleada para el desarrollo de un layout de un estacionamiento es con base a lo sugerido por los expertos Tompkins et al. (2010) en el libro ”Facilities Planning”. Esta indica que hay que seguir los siguientes pasos:

1. Determinar el número de automóviles que son estacionados por tipo de

automóvil.

2. Determinar el espacio requerido por cada automóvil.
3. Determinar el espacio disponible para estacionar.
4. Determinar alternativas de layout de estacionamiento para patrones de layout de estacionamiento.
5. Seleccionar el layout que mejor utiliza el espacio y maximiza la conveniencia del empleado.

Capítulo 2

Marco teórico

2.1. Revisión bibliográfica

Existen estudios que tratan el problema de estacionamientos desde diferentes puntos de vista. Se encontraron los siguientes:

Artículo de Asmael y Turky (2022) *"Parking Requirement of Institutional Land Use"*. Trata sobre el diseño de un estacionamiento usando regresión para determinar el número de plazas que se necesitan para estacionar en áreas urbanas que se encuentran afuera de la ciudad de Baghdad CBD (Asmael y Turky, 2022).

Artículo de Nikolaev et al. (2021) *"Smart parking for an ecological type of transport-electric scooter"*. Se tiene información de cómo diseñar un estacionamiento para scooters. Sin embargo, no se considera este enfoque en scooters para el desarrollo de la metodología planteada al principio debido a que el objetivo del desarrollo del layout del estacionamiento se planea que sea dirigido a audiencias que usen vehículos de mayor tamaño.

Artículo de Leonard et al. (s.f.) *"IA Summary Repoti Employment Center Parking Facilities"*. Es sobre el diseño de un estacionamiento. Se toma en cuenta el cálculo de demanda por categoría de empleo y la alternativa del layout para centros de empleo.

Artículo de McConochie (1961) *"Design of Parking Lots and Garages"*. Trata principalmente de las cocheras. Sin embargo, el autor agrega que "Los principios son generalmente aplicables, de cualquier manera, a los estacionamientos" (McConochie, 1961). Por ello, se puede interpretar que las consideraciones que se toman con las cocheras también pueden ser tomados en cuenta para el diseño de un estacionamiento. Además, ofrece un consejo para aquellos que quieran diseñar un estacionamiento importante o cochera. Este consejo se cita a continuación.

> "El ingeniero o arquitecto debería revisar no solo los manuales estándar y libros de textos en la materia cuando enfrenta la oportunidad de diseñar una coche-

> ra, sino también él debería de examinar críticamente lo más importante de las recientes estructuras".

Lo cual quiere decir que los ingenieros o arquitectos deben revisar la instalación a la cual se le quiere diseñar una cochera para reflexionar y tomar nota de posibles requerimientos estructurales de la obra.

Se encontró otro artículo que hace mención de experiencias de transporte de ingenieros. Al mismo tiempo, sugiere diseños de estacionamientos para uso actual de una instalación sujeta a estudio y, también, para cambios futuros de parámetros. El autor toma de referencia a Board (1971), quien escribió el libro "Principles, Parking", para la metodología. Este último maneja una metodología parecida a la de Tompkins et al. (2010) debido a que considera medidas similares de ancho de cajón de estacionamiento, por ejemplo, de 8, 8.5, 9 y 10 pies. También, ambos consideran los ángulos para el cajón de estacionamiento y las líneas de circulación para la propuesta de layout.

2.2. Estacionamientos

De acuerdo a Tompkins et al. (2010), el layout para un estacionamiento considera los siguientes factores:

1. El porcentaje de automóviles compactos. Como una guía de planeación, el 33 % de todo el estacionamiento es a menudo acomodado para autos compactos.
2. Incrementar el área dada para estacionar disminuye la cantidad de tiempo requerido para estacionar y salir del cajón de estacionamiento.
3. Configuraciones angulares permiten rotación más rápida; estacionar perpendicularmente a menudo da lugar a mayor espacio de utilización, a pesar de que también requiere pasillos más amplios.

2.3. Normas de estacionamiento

Es importante considerar las normas de los estacionamientos. Cada localidad tiene sus propias regulaciones. Incluso, los edificios para el que se realiza el layout tienen sus normas. Por lo que entendemos, es conveniente verificar con las autoridades correspondientes y ver algunos layouts de estacionamientos ya hechos. Es importante saber que, cuando no se cuenta con información sobre autos compactos, el 33 % del estacionamiento se destina a este tipo de automóvil.

El tamaño de cajón puede variar dependiendo del uso que se le dé al estacionamiento. Algunas de las recomendaciones se expresan en la figura 2.1 (Tompkins et al., 2010).

Uso	Ancho (ft)
Uso de automóvi pequeño	7.5 - 8
Uso de estacionamiento de todo el día	8 - 8.5
Uso de automóvil estándar	8.5 - 9
Uso lujuso y para personas mayores	9 - 10
Uso de supermercado y caravana	10 - 11
Uso para personas con discapacidad*	11.875 - 12.125

*Requerimientos mínimos = 1 o 2 por 100 cajones de estacionamiento o lo especificado por ley local, estatal o federal.

Figura 2.1: Tamaño de cajón por uso recomendada (Tompkins et al., 2010)

La Figura 2.1 muestra por ejemplo que para automóviles pequeños o compactos el ancho del cajón de estacionamiento puede ser de hasta 8 ft. Estándares Gráficos Arquitectónicos (AGS por sus siglas en inglés) llega en 1932 como una progresión natural. Esta progresión es parte de un compromiso centenario de John Wiley, quien tuvo un papel en el desarrollo de la arquitectura americana, que publica sobre la calidad en tema de una particular profesión o más profesiones. Ramsey y Sleeper son los autores de la primera edición. Desde entonces,

continúan siendo autores de las siguientes ediciones hasta que Sleeper fallece en el año 1960. Desde el comienzo, con el tiempo, fue adquiriendo buena reputación y hoy en día sirve para saber sobre la reglamentación estándar que se debe usar en el diseño de los estacionamientos. Esta reglamentación ha sido revisada continuamente y documentada en el libro "Architectural Graphic Standardars" (Ramsey y Sleeper, 2011). La publicación más reciente es la doceava edición. Sin embargo, para la presente tesis, se consideran los valores de la onceava edición, edición que muestra datos de grupo de automóvil, ancho de cajón (SW), profundidad de automóvil paralelo a pared (VP^w) o parelelo al enclavamiento (VP^i), tamaño de pasillo (AW) y módulos mínimos (W2 y W4). La clasificación de automóvil se caracteriza por el tamaño de automóvil. El SW es la medición correspondiente al cajón de estacionamiento dependiendo del tamaño de automóvil. Los VP^w y VP^i indican el largo del cajón de estacionamiento, la primera con base a la pared y la segunda con el enclavamiento. La AW es la medición de circulación para los automóviles. La W indica cómo estará distribuido el área del estacionamiento, es decir, si el estacionamiento tiene pasillos, una o dos filas, pared, etcétera. Dichos datos se ilustran en las Figuras 2.2, 2.3 y 2.4. Del mismo modo, Tompkins et al. (2010), quien también toma valores de ancho de módulo, la cual es la distancia del tipo de módulo (W en Figura 2.5), del mismo autor, menciona los mismos datos con la diferencia de que la profundidad de cajón de estacionamiento ahora se abrevia como SD y contiene mayor número de tipos de módulos. Los valores que señala Tompkins et al. (2010) se encuentran en las Figuras 2.5 y 2.6.

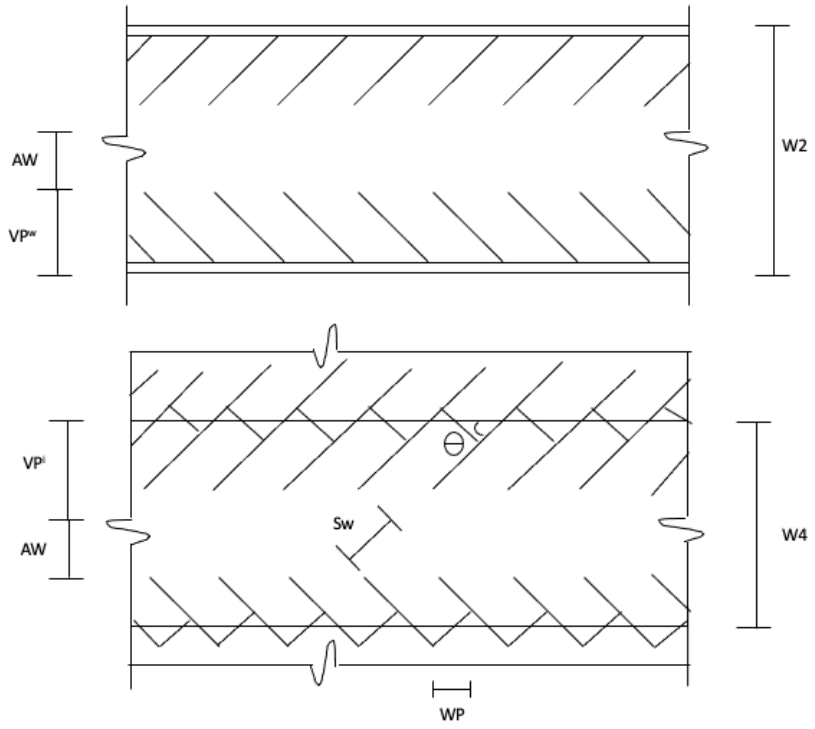


Figura 2.2: Configuración descrita por Ramsey y Sleeper (2011) en el libro *"Arquitectural Graphic Standardar"*.

Ángulo	Tipos de carro	Volumen	Stall Width		Stall Depth paralelo al pasillo		Ancho de pasillo	Módulos mínimos	
			Paralelo a carro	Paralelo al pasillo	A la pared	Al enclavamiento	Mínimo	Pared a pared	o a enclavamiento
			(SW) (FT)	(WP) (FT)	(VP^W) (FT)	(VP^I) (FT)	(AW) (FT)	(W2) (FT)	(W4) (FT)
90	Mixto	A	9	9	18.4	18.4	24	60.8	60.8
		B	8.75	8.8	18.4	18.4	24	60.8	60.8
		C	8.5	8.5	18.4	18.4	24	60.8	60.8
		D	8.25	8.3	18.4	18.4	24	60.8	60.8
	Pequeño	A	8	8	15.1	15.1	22.3	52.4	52.4
		B	7.75	7.8	15.1	15.1	22.3	52.4	52.4
		C	7.5	7.5	15.1	15.1	22.3	52.4	52.4
		D	7.25	7.3	15.1	15.1	22.3	52.4	52.4
75	Mixto	A	9	9.3	19.4	18.6	21	59.9	58.2
		B	8.75	9.1	19.4	18.6	21	59.9	58.2
		C	8.5	8.8	19.4	18.6	21	59.9	58.2
		D	8.25	8.5	19.4	18.6	21	59.9	58.2
	Pequeño	A	8	8.3	16.2	15.4	20	52.5	50.8
		B	7.75	8	16.2	15.4	20	52.5	50.8
		C	7.5	7.8	16.2	15.4	20	52.5	50.8
		D	7.25	7.5	16.2	15.4	20	52.5	50.8
70	Mixto	A	9	9.6	19.5	18.4	16.6	57.5	55.3
		B	8.75	9.3	19.5	18.4	16.6	57.5	55.3
		C	8.5	9	19.5	18.4	16.6	57.5	55.3
		D	8.25	8.8	19.5	18.4	16.6	57.5	55.3
	Pequeño	A	8	8.5	16.4	15.3	17.9	50.6	48.4
		B	7.75	8.2	16.4	15.3	17.9	50.6	48.4
		C	7.5	8	16.4	15.3	17.9	50.6	48.4
		D	7.25	7.7	16.4	15.3	17.9	50.6	48.4
65	Mixto	A	9	9.9	19.4	18	16.1	54.9	52.2
		B	8.75	9.7	19.4	18	16.1	54.9	52.2
		C	8.5	9.4	19.4	18	16.1	54.9	52.2
		D	8.25	9.1	19.4	18	16.1	54.9	52.2
	Pequeño	A	8	8.8	16.4	15	15.7	48.5	45.8
		B	7.75	8.6	16.4	15	15.7	48.5	45.8
		C	7.5	8.3	16.4	15	15.7	48.5	45.8
		D	7.25	8	16.4	15	15.7	48.5	45.8

Figura 2.3: Ancho de módulo por Ramsey y Sleeper (2011) en el libro *"Arquitectural Graphic Standardar"*.

Ángulo	Tipos de carro	Volumen	Stall Width		Stall Depth paralelo al pasillo		Ancho de pasillo	Módulos mínimos	
			Paralelo a carro	Paralelo al pasillo	A la pared	Al enclavamiento	Mínimo	Pared a pared	Enclavamiento a enclavamiento
			(SW) (FT)	(WP) (FT)	(VP^W) (FT)	(VP^I) (FT)	(AW) (FT)	(W2) (FT)	(W4) (FT)
60	Mixto	A	9	10.4	19.1	17.5	11.7	51.9	48.7
		B	8.75	10.3	19.1	17.5	11.7	51.9	48.7
		C	8.5	9.8	19.1	17.5	11.7	51.9	48.7
		D	8.25	9.5	19.1	17.5	11.7	51.9	48.7
	Pequeño	A	8	9.2	16.3	14.7	13.6	46.1	42.9
		B	7.75	8.9	16.3	14.7	13.6	46.1	42.9
		C	7.5	8.7	16.3	14.7	13.6	46.1	42.9
		D	7.25	8.4	16.3	14.7	13.6	46.1	42.9
55	Mixto	A	9	11	18.7	16.9	11.2	48.7	45.1
		B	8.75	10.7	18.7	16.9	11.2	48.7	45.1
		C	8.5	10.4	18.7	16.9	11.2	48.7	45.1
		D	8.25	10.1	18.7	16.9	11.2	48.7	45.1
	Pequeño	A	8	9.8	16	14.2	11.5	43.5	39.8
		B	7.75	9.6	16	14.2	11.5	43.5	39.8
		C	7.5	9.2	16	14.2	11.5	43.5	39.8
		D	7.25	8.9	16	14.2	11.5	43.5	39.8
50	Mixto	A	9	11.7	18.2	16.2	11	47.4	43.3
		B	8.75	11.4	18.2	16.2	11	47.4	43.3
		C	8.5	11.1	18.2	16.2	11	47.4	43.3
		D	8.25	10.8	18.2	16.2	11	47.4	43.3
	Pequeño	A	8	10.4	15.7	13.6	11	42.4	38.2
		B	7.75	10.1	15.7	13.6	11	42.4	38.2
		C	7.5	9.8	15.7	13.6	11	42.4	38.2
		D	7.25	9.5	15.7	13.6	11	42.4	38.2
45	Mixto	A	9	12.7	17.5	15.3	11	46.1	41.5
		B	8.75	12.4	17.5	15.3	11	46.1	41.5
		C	8.5	12	17.5	15.3	11	46.1	41.5
		D	8.25	11.7	17.5	15.3	11	46.1	41.5
	Pequeño	A	8	11.3	15.2	12.9	11	41.4	36.9
		B	7.75	11	15.2	12.9	11	41.4	36.9
		C	7.5	10.6	15.2	12.9	11	41.4	36.9
		D	7.25	10.3	15.2	12.9	11	41.4	36.9

Figura 2.4: Ancho de módulo por Ramsey y Sleeper (2011) en el libro *"Arquitectural Graphic Standardar"*.

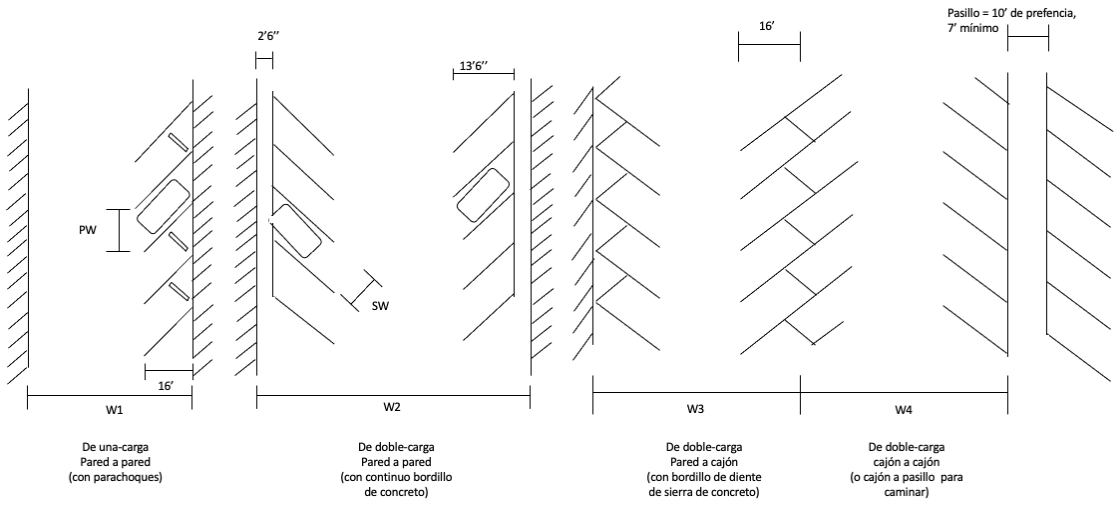


Figura 2.5: Tipo de módulo de estacionamiento descrito por Tompkins et al. (2010) en el libro *"Facilities Planning"*.

			Ángulo de estacionamiento								
			45	50	55	60	65	70	75	80	85	90
Grupo I: carros pequeños	8'0"	1	25'9"	26'6"	27'2"	29'4"	31'9"	34'0"	36'2"	38'2"	40'0"	41'9"
		2	40'10"	42'0"	43'1"	45'8"	48'2"	50'6"	52'7"	54'4"	55'11"	57'2"
		3	38'9"	40'2"	41'5"	44'2"	47'0"	49'6"	51'10"	53'10"	55'8"	57'2"
		4	36'8"	38'3"	39'9"	42'9"	45'9"	48'6"	51'1"	53'4"	55'5"	57'2"
Grupo II: carros estándar	8'6"	1	32'0"	32'11"	34'2"	36'2"	38'5"	41'0"	43'6"	45'6"	46'11"	48'0"
		2	49'10"	51'9"	53'10"	56'0"	58'4"	60'2"	62'0"	63'6"	64'9"	66'0"
		3	47'8"	49'4"	51'6"	54'0"	56'6"	59'0"	61'2"	63'0"	64'6"	66'0"
		4	45'3"	46'10"	49'0"	51'8"	54'6"	57'10"	60'0"	62'6"	64'3"	66'0"
Grupo III: carros largos	9'0"	1	32'0"	32'9"	34'0"	35'4"	37'6"	39'8"	42'0"	44'4"	46'2"	48'0"
		2	49'4"	51'0"	53'2"	55'6"	57'10"	60'0"	61'10"	63'4"	64'9"	66'0"
		3	46'4"	48'10"	51'4"	53'10"	56'0"	58'8"	61'0"	63'0"	64'6"	66'0"
		4	44'8"	46'6"	49'0"	51'6"	54'0"	57'0"	59'8"	62'0"	64'2"	66'0"
	9'6"	1	32'0"	32'8"	34'0"	35'0"	36'10"	38'10"	41'6"	43'8"	46'0"	48'0"
		2	49'2"	50'6"	51'10"	53'6"	55'4"	58'0"	60'6"	62'8"	64'6"	65'11"
		3	47'0"	48'2"	49'10"	51'6"	53'1"	57'0"	59'8"	62'0"	64'3"	65'11"
		4	44'8"	45'10"	47'6"	49'10"	52'6"	55'9"	58'9"	61'6"	63'10"	65'11"
	9'0"	1	32'7"	33'0"	34'0"	35'11"	38'3"	40'11"	43'6"	45'5"	46'9"	48'0"
		2	50'2"	51'2"	53'3"	55'4"	58'0"	60'4"	62'9"	64'3"	65'5"	66'0"
		3	47'9"	49'1"	52'3"	53'8"	56'2"	59'2"	61'11"	63'9"	65'2"	66'0"
		4	45'5"	46'11"	49'0"	51'8"	54'9"	58'0"	61'0"	63'2"	64'10"	66'0"
	9'6"	1	32'4"	32'8"	33'10"	34'11"	37'2"	39'11"	42'5"	45'0"	46'6"	48'0"
		2	49'11"	50'11"	52'2"	54'0"	56'6"	59'3"	61'9"	63'4"	64'8"	66'0"
		3	47'7"	48'9"	50'2"	52'4"	55'1"	58'4"	60'11"	62'10"	64'6"	66'0"
		4	45'5"	46'8"	48'5"	50'8"	55'8"	57'0"	59'10"	52'2"	64'1"	66'0"
	10'0"	1	32'4"	32'8"	33'10"	34'11"	37'2"	39'11"	42'5"	45'0"	46'6"	48'0"
		2	49'11"	50'11"	52'2"	54'0"	56'6"	59'3"	61'9"	63'4"	64'8"	66'0"
		3	57'7"	48'9"	50'2"	52'4"	55'1"	58'4"	60'11"	62'11"	64'6"	66'0"
		4	45'3"	46'8"	48'5"	50'8"	53'8"	57'0"	59'10"	62'2"	64'1"	66'0"

Figura 2.6: Ancho de módulo por Tompkins et al. (2010) en el libro ”Facilities Planning”.

En las Figuras 2.3, 2.4 y 2.6 se puede obsevar que el ancho de módulo es similar aunque no igual. Por ejemplo, suponiendo que se tiene un cajón de estacionamiento de 90º, tipo de módulo W4 (W) y tamaño de automóvil pequeños con ancho de cajón de estacionamiento equivalente a 8 ft, el ancho de módulo debería de ser igual a 52.4 ft según Ramsey y Sleeper (2011). Mientras Tompkins et al. (2010) con los mismo datos indica que debería de ser de 57.1666 ft. Se recomienda utilizar la referencia de la Figura 2.6 ya que proporciona información del ancho de módulo para los cuatro tipos de módulos: W1, W2, W3 y W4. Mientras que las Figuras 2.3 y 2.4 muestra solo información del W2 y W4.

De acuerdo a la NORMA Oficial Mexicana NOM-030-SSA3-2013 (Secretaría de gobernación, 2020) y la Norma Mexicana NMX-R-050-SCFI-2006 (Secretaría de gobernación, 2006), las regulaciones de estacionamiento para personas con discapacidad son colocar el correspondiente al 4 % del total de cajones o como mínimo un cajón, si este porcentaje no es alcanzado. Las dimensiones deben ser de 3.8 m de ancho por 5 m de largo. Al convertilo a pies es de 12.4672 por 16.4042, respectivamente. En Celaya, las normas de las dimensiones

de los vehículos deben ser de 2.5 x 5 m. En pies, es igual a 8.2021 x 16.4042 ft. En el caso de autobuses el ancho mínimo es de 3.5 m, es decir, 11.4829 ft. También se hace mención de que estas dimensiones cumplen con las regulaciones de las NOM y NRM (*Normas técnicas complementarias del regramento de ordenamiento territorial (ROT) del municipio de Celaya, Gto. para el proyecto arquitectónico*, 2006). Por lo tanto, las normas mexicanas son parecidas en dimensiones a las americanas. Sin embargo, se recomienda revisar las normas de acuerdo a la zona en que se desea implementar el layout.

Capítulo 3

Metodología

3.1. Metodología para el diseño de un estacionamiento

La metodología está dividida en cinco secciones para el diseño de un estacionamiento. En la Figura 3.1 se explica el funcionamiento del algoritmo. Para propósitos de este estudio, el algoritmo se programa con el software Python. El código se describe y se muestra en los apéndices A y B.

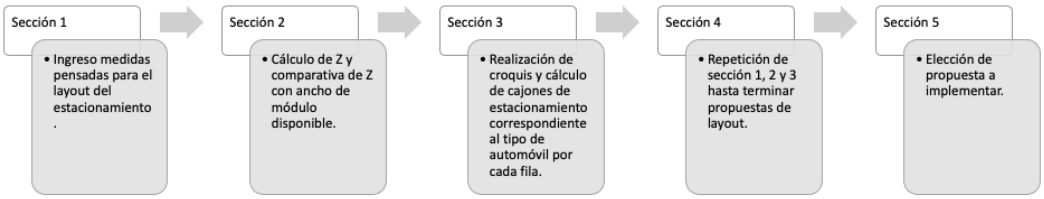


Figura 3.1: Descripción de la metodología.

Sección 1. Primero, se le pide al usuario ingresar la cantidad de alternativas de layouts. Segundo, los parámetros de demanda total (cantidad de automóviles que usan el estacionamiento) y orientación de espacio (vertical u horizontal) son los siguiente datos a ingresar. Algunos ejemplos de orientación de espacio vertical se muestran en la Figura 2.5. Se entiende como espacio vertical porque las filas de estacionamiento se encuentran perpendicular al ancho del módulo. Posteriormente, se le pregunta al usuario si requiere cajón de estacionamiento para personas con discapacidad. Si el usuario confirma que sí, se le pide al usuario nuevamente ingresar los datos: porcentaje de automóviles con discapacidad con respecto a la demanda total, escoger el ancho de cajón de estacionamiento de los automóviles, profundidad de cajón de estacionamiento, ancho de cajón de estacionamiento respecto al estacionamiento, tipo de módulo y ángulo. Tercero, en el programa se visualiza el valor correspondiente a los datos ingresados previamente. Además, muestra valores de PW (ancho de módulo con ángulo y profundidad), número de módulos y profundidad de módulo requerido. Cuarto, una vez terminado el proceso con formulas y cálculos (ver ejemplos en la sección 3.2 de la presente

tesis) con los cajones para personas con discapacidad se repite el procedimiento para cada uno de los tipos de autos.

```
Table. Parking layout alternative ----------------------------------------------
Demand (units):  23.0
Parking angle (º):  90.0
Module type (W):  W1
SW (feet):  8.0
PW (feet):  8.0
y=  1.0446482126086753e-15
x=  28.0
Horizontal # modules:  1.0
Module Depth required =  224.0
Compact car stalls per module =  28.0
Module Width (feet):  41.75
Standard? (yes/no) no
Module Depth required =  0.0
Handicapped car stalls per module =  0.0
Module Width (feet):  0.0
Large? (yes/no) no
Module Depth required (feet) =  0.0
Large car stalls per module =  0.0
Module Width (feet):  0.0
Z =  41.75
```

Figura 3.2: Resumen de datos de sección 1.

Sección 2. En esta parte, se determina el número de cajones de estacionamiento por módulo. En caso de que el estacionamiento no necesite algún tipo de automóvil, el número de módulos, profundidad de módulos y espacios por módulo son igual a 0. Posteriormente, se realizará el cálculo de Z para revisar si es factible el layout del estacionamiento. Z corresponde a la distancia requerida para cubrir los módulos necesarios para el layout del estacionamiento (Figura 3.3). Si Z es menor o igual al ancho del terreno del estacionamiento, entonces es “Factible”.

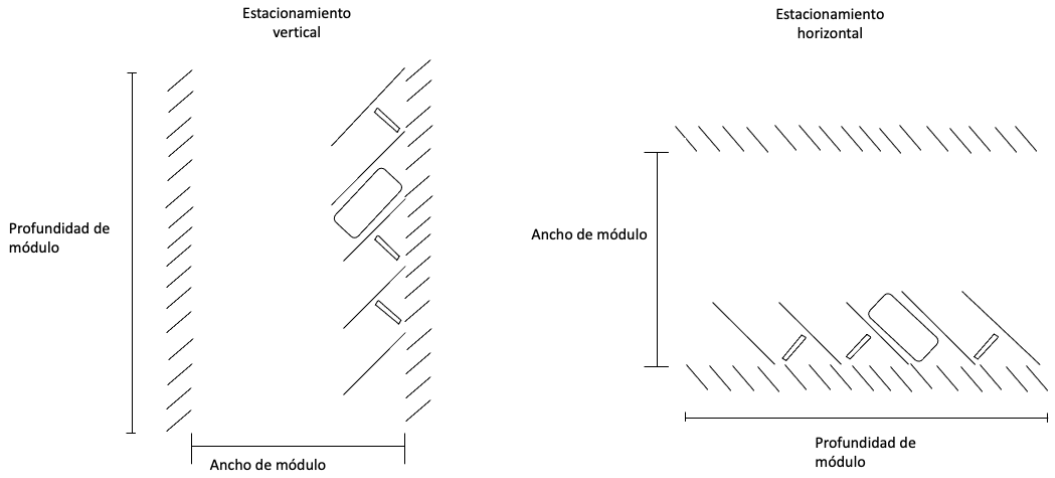


Figura 3.3: Descripción de módulos vertical y horizontal.

Sección 3. Si la respuesta es ”Factible”, se muestran imágenes (Figuras 3.4, 3.5, 3.6 y 3.7) junto con instrucciones para que el usuario pueda realizar el croquis por su propia cuenta. Luego, calcula el total de filas de cajones de estacionamiento y el programa pregunta si existen espacios para discapacitados y líneas de circulación para cada una de las filas de cajones de estacionamiento. Estos espacios pueden ser observados en el croquis previo realizado por el usuario. Considerando los cajones de estacionamiento y espacios de circulación, se calcula la cantidad de cajones de estacionamiento correspondiente al tipo de automóvil por cada fila. Dichos cálculos se ejemplifican en el apartado siguiente de Resultados. Después, existe un resumen de los resultados del algoritmo.

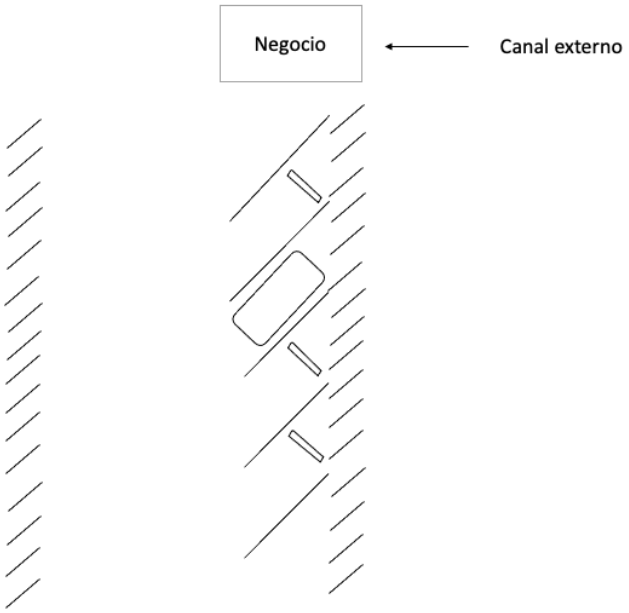


Figura 3.4: Imagen “channels” del algoritmo.

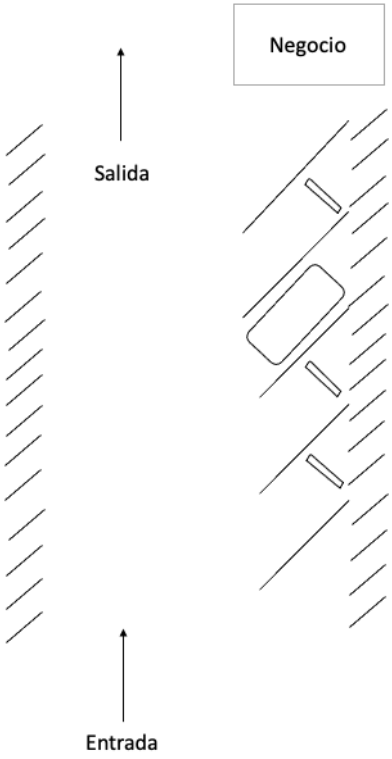


Figura 3.5: Imagen “entrance and exit” del algoritmo.

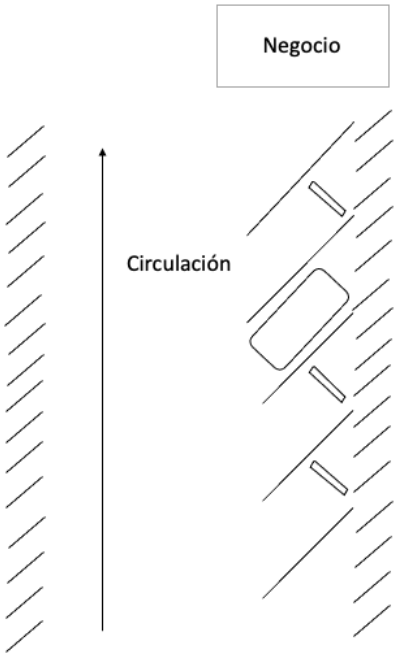


Figura 3.6: Imagen “circulation lines” del algoritmo.

Figura 3.7: Imagen “handicapped spaces” del algoritmo.

Sección 4. Se continúa con el siguiente número de alternativa de layout que ingresa en

principio el usuario.

Sección 5, el usuario selecciona la alternativa de estacionamiento que considere más conveniente implementar. Para ello, se tiene que revisar cuál de los layouts cubre todos los requerimientos del estacionamiento. Por ejemplo, un requerimiento puede ser que el layout propuesto cumpla con la demanda de cajones de estacionamiento por tipo de auto. La aplicación del programa desarrollada en Python, y los resultados, se describen en el siguiente apartado.

3.2. Resultados

Los resultados consisten en comparar los layouts de cinco estacionamientos construidos en Ciudad del Carmen, Campeche, México contra los obtenidos con la metodología de Tompkins et al. (2010), programada en Python.

3.2.1. Estacionamiento 1

Se tomaron medidas del estacionamiento del mercado Alonso Felipe de Andrade de Ciudad del Carmen, Campeche. Las medidas reales se muestran en la Figura 3.8.

Ubicación: Calle 20, Centro, 24100 Cd. del Carmen, Camp.

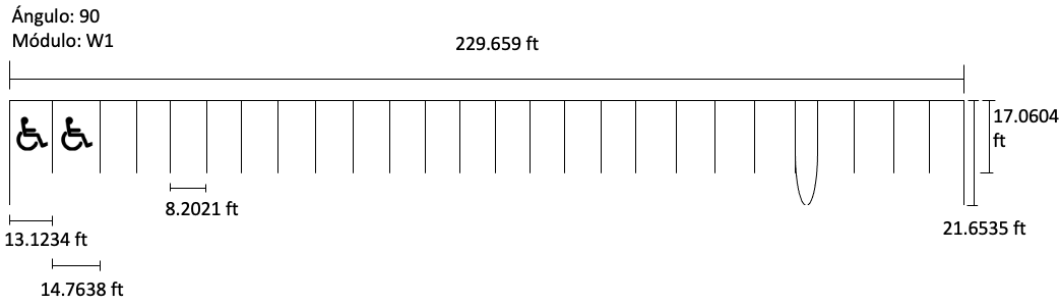


Figura 3.8: Croquis original de estacionamiento 1.

Para el cálculo de PW, número de módulos horizontales y Z se usan las siguientes fórmu-

	Layout original		Para generación de layout	
	Compacto	Personas con discapacidad	Compacto	Personas con discapacidad
Demanda	23	2	23	2
Ángulo	90	90	90	90
Tipo de módulo	W1	W1	W1	W1
SW (ft)	8.2021	13.1234 y 14.7638	8	14.7638
PW (ft)	8.2021	13.1234 y 14.7638	8	14.7638
# módulos horizontales	1		No factible	
Espacio disponible (ft)	21.6536		21.6536	
Z			41.75	

Figura 3.9: Comparación de datos de entrada para el estacionamiento 1.

las:

$$y = SD\cos\theta \tag{3.1}$$

$$PW = \frac{SW}{\sin\theta} \tag{3.2}$$

$$Número de cajones por módulo = \frac{Profundidad\ de\ módulo - y}{PW} * número de\ filas \tag{3.3}$$

$$Número\ de\ módulos\ horizontales\ o\ verticales = \frac{Demanda}{Cajones\ por\ módulo} \tag{3.4}$$

$$Z = Número de módulos horizontal o vertical * Ancho de módulo respecto a tabla \tag{3.5}$$

$$Fila = \frac{Profundidad\ de\ módulo - y - Espacio\ para\ discapacidad - Circulación}{PW} \tag{3.6}$$

Se representa con la “y” el ancho que ocupa la profundidad respecto al ángulo. Se puede observar en la Figura 3.10.

Al aplicar las fórmulas, para el layout generado del estacionamiento 1, los resultados son los siguientes

$$y = 17.0604'\cos 90^\circ = 0' \tag{3.7}$$

$$PW = \frac{8'}{\sin 90^\circ} = 8' \tag{3.8}$$

$$Número\ de\ cajones\ por\ módulo = \frac{229.659' - 0'}{8'} * 1 = 27.7073 \approx 27 \tag{3.9}$$

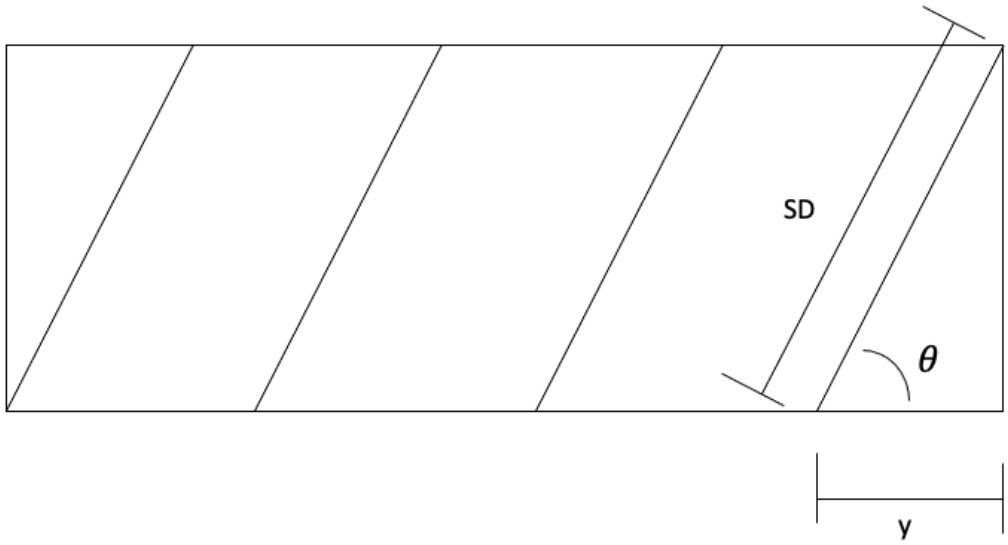


Figura 3.10: Lugar de “y”.

$$Número\ de\ módulos\ horizontales\ para\ cajones\ compactos = \frac{25}{27} = 0.9259 \approx 1 \quad (3.10)$$

$$Z = 1 * 41.75' = 41.75' \quad (3.11)$$

Al obtener el ancho del espacio disponible del estacionamiento (21.6536 ft) menor a Z (41.75 ft). El algoritmo arroja que no es factible. Por lo tanto, no es posible implementar el layout desarrollado con el programa.

3.2.2. Estacionamiento 2

Por razones de confidencialidad, no se indica la ubicación del estacionamiento. Las medidas reales se muestran en la Figura 3.11. El estacionamiento corresponde a una farmacia. Se compara el layout original (Figura 3.11) con el que genera el programa (Figura 3.12).

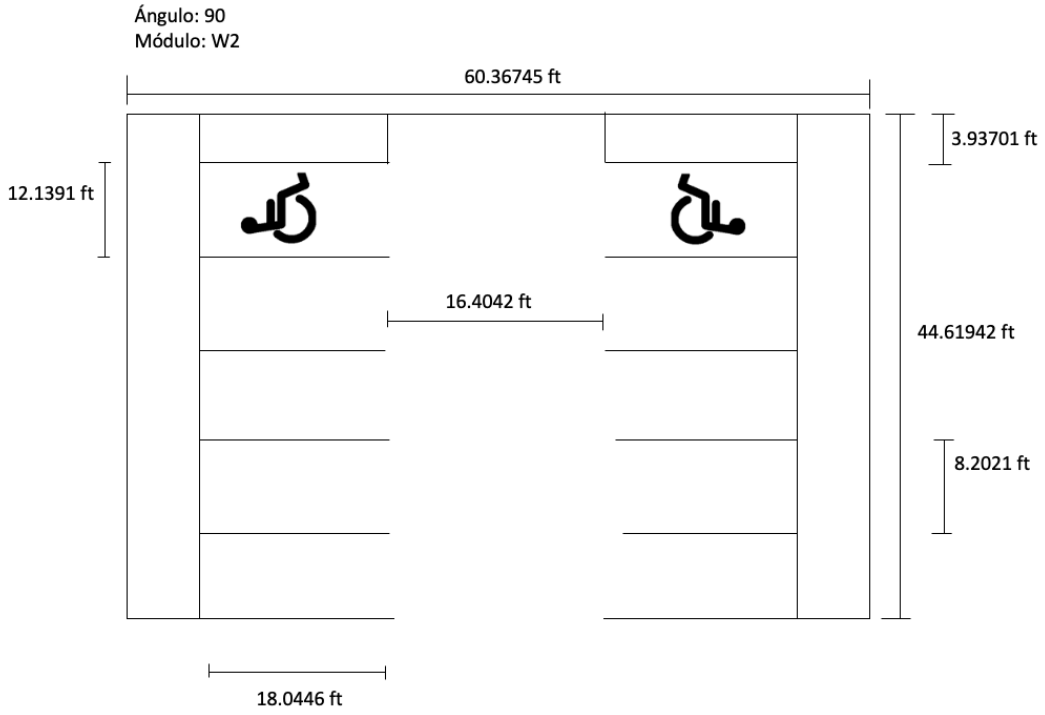


Figura 3.11: Croquis original de estacionamiento 2.

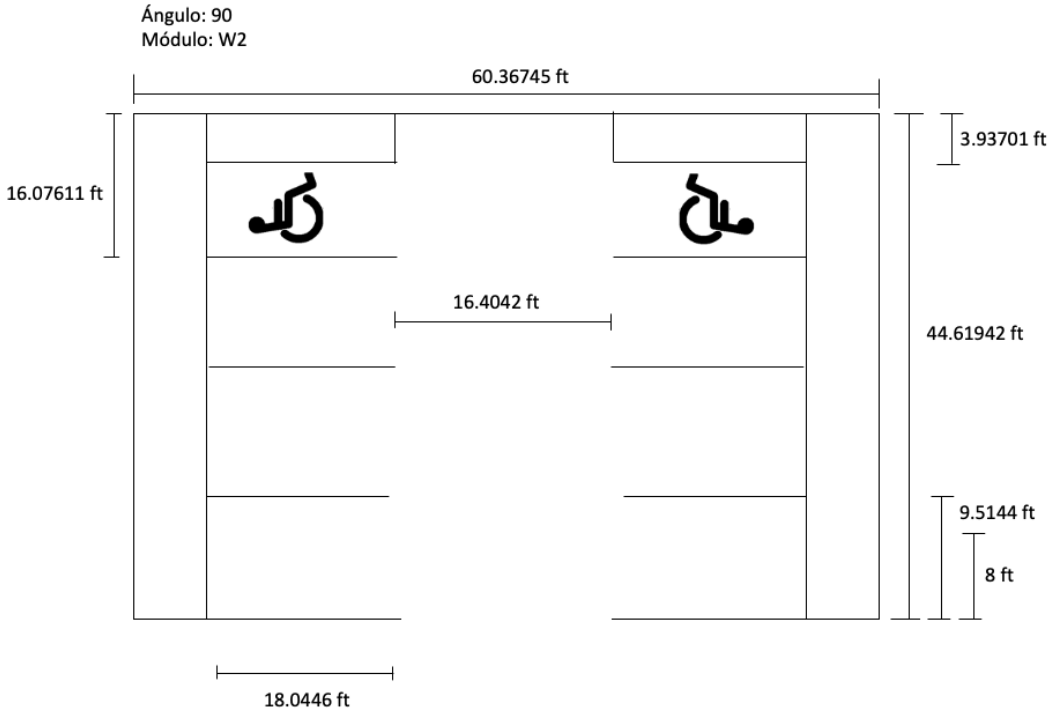


Figura 3.12: Croquis generado de estacionamiento 2.

Al aplicar el algoritmo al estacionamiento 2 se consideran estos valores:

	Layout original		Para generación de layout	
	Compacto	Personas con discapacidad	Compacto	Personas con discapacidad
Demanda	8	2	8	2
Ángulo	90	90	90	90
Tipo de módulo	W2	W2	W2	W2
SW (ft)	8.2021	12.1391 (sin rampa)	8	16.07611 (con rampa)
PW (ft)	8.2021	12.1391	8	16.07611 (con rampa)
# módulos verticales	1		1	
Espacio disponible (ft)	44.61942		44.61942	
Z			57.16666	

Figura 3.13: Comparación de datos de entrada para el estacionamiento 2

Aplicando las fórmulas 3.1, 3.2, 3.3, 3.3, 3.4 y 3.5 los resultados quedan de la siguiente manera:

$$y = 18.0446' \cos 90^\circ = 0 \approx 0' \tag{3.12}$$

$$PW = \frac{8'}{\sin 90^\circ} = 8' \tag{3.13}$$

$$Número\ de\ cajones\ por\ módulo = \frac{44.61942' - 0'}{8'} * 2 = 11.1548 \approx 11 \tag{3.14}$$

$$Número\ de\ módulos\ horizontales\ para\ cajones\ compactos = \frac{10}{11} = 0.9090 \approx 1 \tag{3.15}$$

$$Z = 1 * 57.1666' = 57.1666' \tag{3.16}$$

Al tener un espacio disponible mayor a Z, se dice que es factible implementar el layout. En este caso, el layout que resulta es como se muestra en la Figura 3.12. Después, se obtienen los resultados de cantidad de cajones de estacionamiento para automóviles de tipo compacto de cada fila:

	Layout original		Layout generado	
	Compactos	Personas con discapacidad	Compactos	Personas con discapacidad
Fila #1	4	1	3	1
Fila #2	4	1	3	1
Total de cajones	8	2	6	2
Demanda	8	2	8	2

Figura 3.14: Comparación de cantidad de cajones de estacionamiento por fila para el estacionamiento 2

$$Fila\ 1 = \frac{44.61942' - 0' - 16.07611 - 0'}{8'} = 3.5679 \approx 3 \tag{3.17}$$

$$Fila\ 2 = \frac{44.61942' - 0' - 16.07611 - 0'}{8'} = 3.5679 \approx 3 \tag{3.18}$$

Lo siguiente es comparar los resultados del layout original con el generado y observar qué tan parecidos son y cuáles son sus desventajas y ventajas.

	Layout original	Layout generado
Desventajas	El ángulo de 90 genera dificultad para estacionarse ya que las medidas del ancho son justas.	Los cajones de estacionamiento no cubren la demanda.
Ventajas	El ángulo de 90 aprovecha el espacio del estacionamiento.	Hay mayor facilidad de estacionamiento ya que, aunque los cajones de estacionamiento tiene un ángulo de 90, el ancho de cajón de estacionamiento es mayor al original.

Figura 3.15: Comparación de desventajas y ventajas de los layouts para el estacionamiento 2

Se puede observar que los layouts sí son parecidos porque cuentan con el mismo número de espacios para personas con discapacidad, misma profundidad y ancho de módulo. La diferencia está en la medida del cajón de estacionamiento debido a que es más grande que la original. Por lo tanto, no genera el mismo número de cajones de estacionamiento (como se muestra en la Figura 3.14).

3.2.3. Estacionamiento 3

Se tomaron medidas del estacionamiento público de Playa Norte. Las medidas reales se muestran en la Figura 3.16.

Ubicación: P.º del Mar, Justo Sierra, 24114, Cd. del Carmen, Camp.

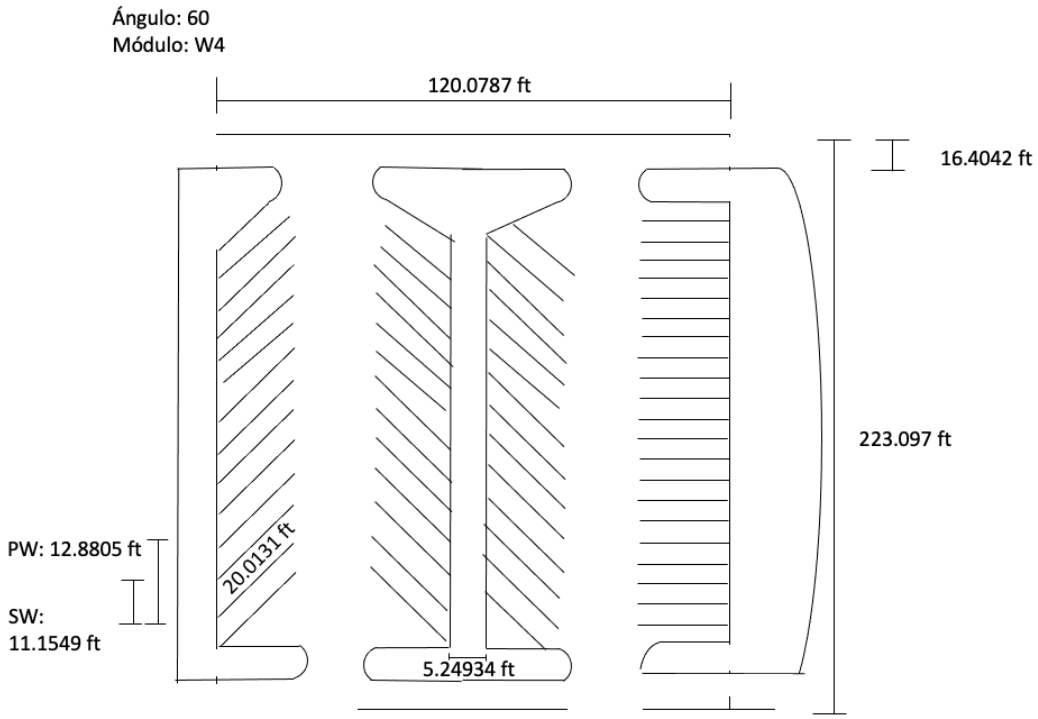


Figura 3.16: Croquis original del estacionamiento 3

Al aplicar el algoritmo se agregan los valores de la siguiente manera.

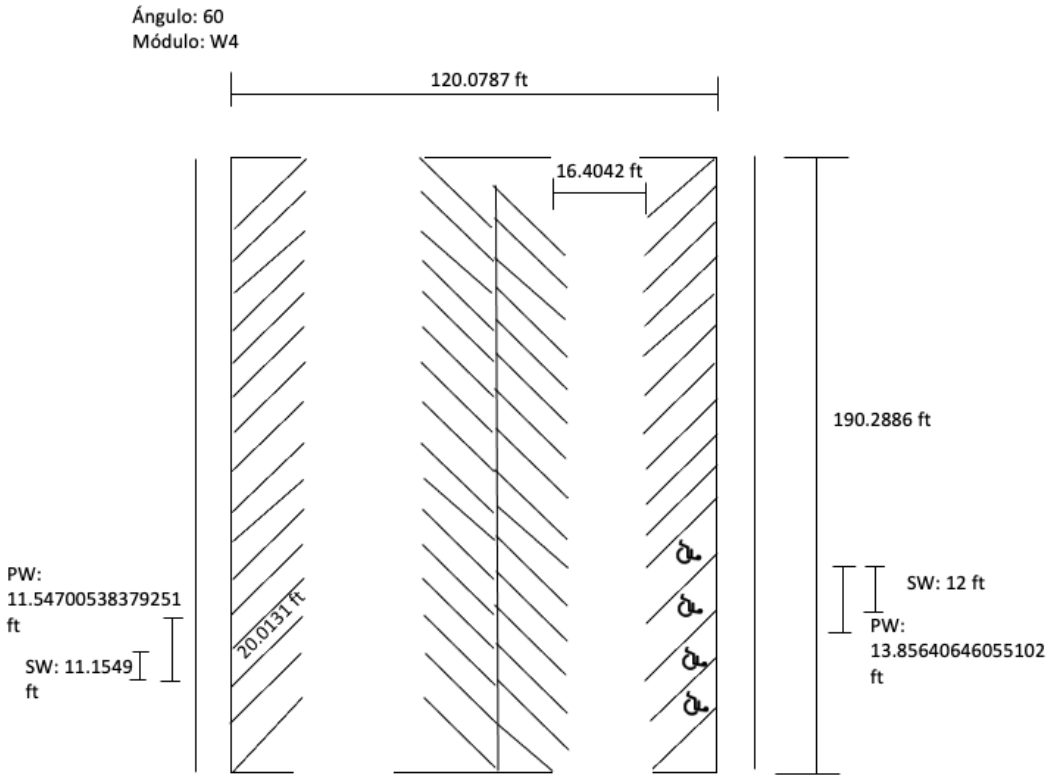


Figura 3.17: Croquis generado del estacionamiento 3 para automóviles largos.

	Layout original		Para generación de layout	
	Largo	Personas con discapacidad	Compacto	Personas con discapacidad
Demanda	61		58	4
Ángulo	60		60	60
Tipo de módulo	W4		W4	W4
SW (ft)	11.1549		10	12
PW (ft)	12.8805		11.547	13.8564
# módulos horizontales	2		2	
Espacio disponible (ft)	120.0787		120.0787	
Z			101.3333	

Figura 3.18: Comparación de datos de entrada para el estacionamiento 3 con automóviles largos.

$$y = 20.0131' \cos 60^\circ = 10.0065' \tag{3.19}$$

$$PW = \frac{10'}{\sin 60^\circ} = 11.5470' \tag{3.20}$$

$$Número\ de\ cajones\ por\ módulo = \frac{190.2886' - 10.0065'}{11.5470'} * 2 = 31.2257 \approx 31 \tag{3.21}$$

$$Número\ de\ módulos\ horizontales\ para\ cajones\ compactos = \frac{61}{31} = 1.96774 \approx 2 \tag{3.22}$$

$$Z = 2 * 58.6666' = 101.3333' \tag{3.23}$$

El espacio disponible (120.0787 ft) alcanza para que sea factible implementar el layout, debido a que es menor Z (101.3333 ft). Por lo tanto, se procede a lo siguiente.

	Layout original		Layout generado	
	Largos	Personas con discapacidad	Largos	Personas con discapacidad
Fila #1	22	0	6	4
Fila #2	13	0	15	0
Fila #3	13	0	15	0
Fila #4	13	0	15	0
Total de cajones	61	0	51	4
Demanda	61	0	61	0

Figura 3.19: Comparación de cantidad de cajones de estacionamiento por fila para el estacionamiento 3 para automóviles largos.

$$Fila\ 1 = \frac{190.2886' - 10.0065' - 4 * 13.8564 - 0'}{11.5470'} = 10.8128 \approx 10. \tag{3.24}$$

$$Fila\ 2 = \frac{190.2886' - 10.0065' - 0' - 0'}{11.5470'} = 15.6128 \approx 15 \tag{3.25}$$

$$Fila\ 3 = \frac{190.2886' - 10.0065' - 0' - 0'}{11.5470'} = 15.6128 \approx 15 \tag{3.26}$$

$$Fila\ 4 = \frac{190.2886' - 10.0065' - 0' - 0'}{11.5470'} = 15.6128 \approx 15 \tag{3.27}$$

	Layout original	Layout generado
Desventajas	No hay espacios exclusivos para personas con discapacidad.	Los cajones de estacionamiento no cumplen con la medida para automóviles largos del layout original.
Ventajas	El estacionamiento cubre la demanda y cumple con la medida para automóviles largos.	Hay espacios de automóviles para personas con discapacidad.

Figura 3.20: Comparación de desventajas y ventajas de los layouts para el estacionamiento 3 para automóviles largos.

Dado que el layout del estacionamiento no cubre la demanda total, se decide cambiar el SW de 10ft a uno de un auto compacto igual a 8 ft para observar si es factible. En este caso, sí lo es. Así que el croquis se observa en la Figura 3.21.

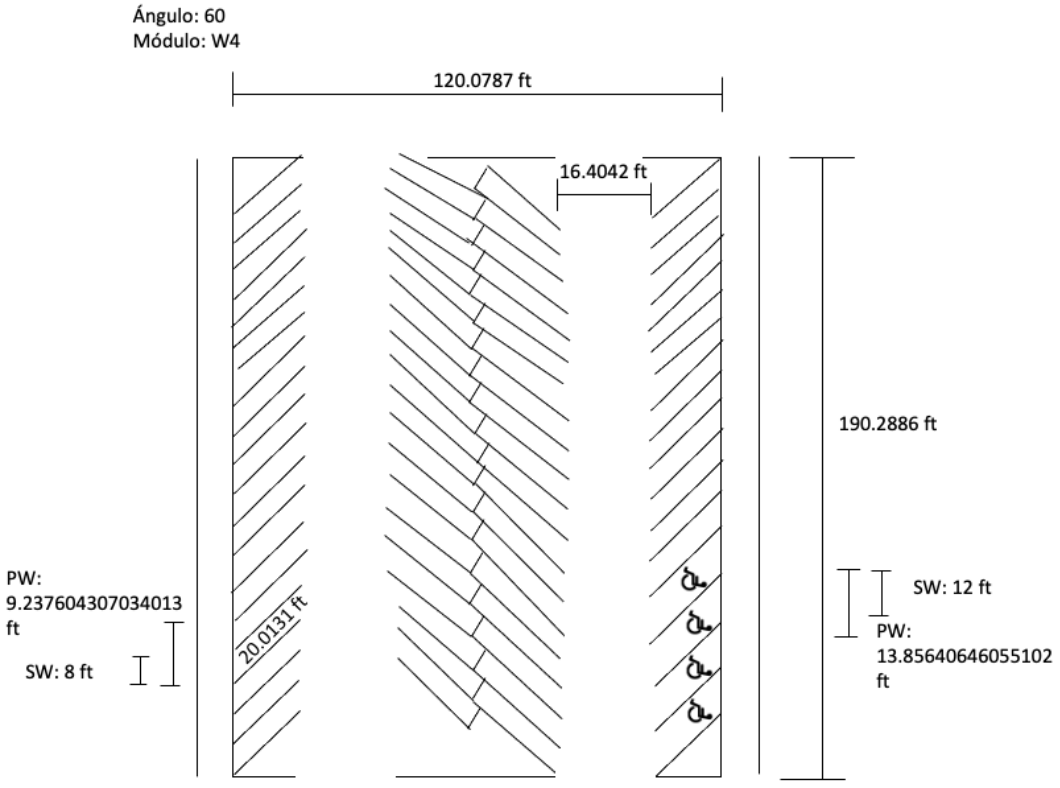


Figura 3.21: Croquis generado del estacionamiento 3.

Los valores quedan de la siguiente manera.

	Layout original		Para generación de layout	
	Largo	Personas con discapacidad	Compacto	Personas con discapacidad
Demanda	61		58	4
Ángulo	60		60	60
Tipo de módulo	W4		W4	W4
SW (ft)	11.1549		8	12
PW (ft)	12.8805		9.23760431	13.8564
# módulos horizontales	2		2	
Espacio disponible (ft)	120.0787		120.0787	
Z			85.5	

Figura 3.22: Comparación para el estacionamiento 3

$$y = 20.0131' \cos 60^\circ = 10.0065' \tag{3.28}$$

$$PW = \frac{8'}{\sin 60^\circ} = 9.2376' \tag{3.29}$$

$$Número\ de\ cajones\ por\ módulo = \frac{190.2886' - 10.0065'}{9.2376'} * 2 = 38.971 \approx 38 \tag{3.30}$$

$$Número\ de\ módulos\ horizontales\ para\ cajones\ compactos = \frac{61}{38} = 1.6052 \approx 2 \tag{3.31}$$

$$Z = 2 * 42.75' = 85.5' \tag{3.32}$$

Como se puede observar en la Figura 3.13, el layout es factible (Z menor o igual al espacio disponible). Además, cabe mencionar que se modifica acorde al 5 % de la demanda para espacios para personas con discapacidad en una zona que facilite el acceso al malecón costero de Ciudad del Carmen.

	Layout original		Layout generado	
	Largos	Personas con discapacidad	Compactos	Personas con discapacidad
Fila #1	22	0	13	4
Fila #2	13	0	19	0
Fila #3	13	0	19	0
Fila #4	13	0	19	0
Total de cajones	61	0	70	4
Demanda	61	0	61	0

Figura 3.23: Comparación de cantidad de cajones de estacionamiento por fila para el estacionamiento 3

$$Fila\,1 = \frac{190.2886' - 10.0065' - 4 * 9.2376 - 0'}{9.2376'} = 13.5161 \approx 13 \tag{3.33}$$

$$Fila\,2 = \frac{190.2886' - 10.0065' - 0 - 0'}{9.2376'} = 19.5161 \approx 19 \tag{3.34}$$

$$Fila\,3 = \frac{190.2886' - 10.0065' - 0 - 0'}{9.2376'} = 19.5161 \approx 19 \tag{3.35}$$

$$Fila\,4 = \frac{190.2886' - 10.0065' - 0 - 0'}{9.2376'} = 19.5161 \approx 19 \tag{3.36}$$

	Layout original	Layout generado
Desventajas	No hay espacios exclusivos para personas con discapacidad.	Los cajones de estacionamiento no cumplen con la medida para automóviles largos.
Ventajas	El estacionamiento cubre la demanda y cumple con la medida para automóviles largos.	Hay espacios de automóviles para personas con discapacidad. El estacionamiento subre la demanda.

Figura 3.24: Comparación de desventajas y ventajas de los layouts para el estacionamiento 3

El croquis generado se parece al original porque tiene el mismo módulo y número de filas. Sin embargo, se diferencia en el tipo de automóviles que pueden estacionarse. Además,

cubre el uso exclusivo mínimo para personas con discapacidad (4 %) de acuerdo a las normas mexicanas.

3.2.4. Estacionamiento 4

Se tomaron medidas del estacionamiento público de Playa Norte. Las medidas reales se muestran en la Figura 3.16.

Ubicación: P.º del Mar, Justo Sierra, 24114, Cd. del Carmen, Camp. El número de cajones para discapacitados es igual a 2. Sin embargo, este número se modifica para la propuesta de layout ya que, del marco teórico se sabe que, el 4 % de la demanda total debe de estar destinado a personas con dicapacidad. Se decide usar el 5 %.

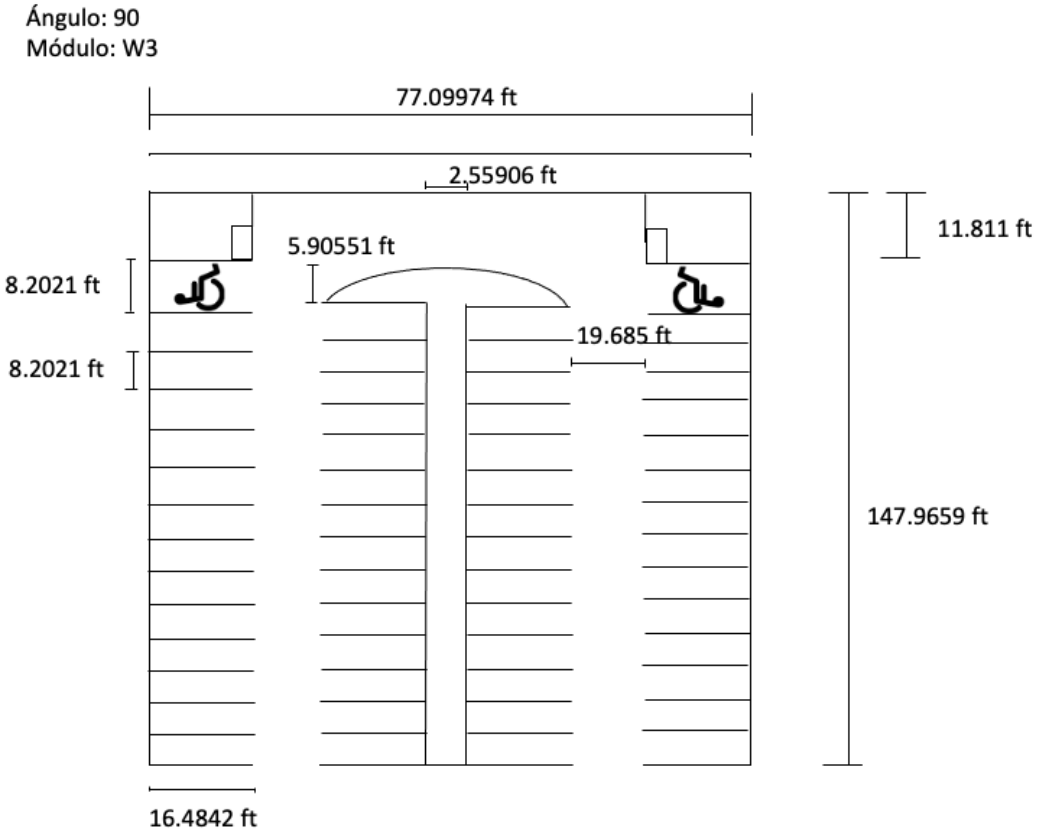


Figura 3.25: Croquis original del estacionamiento 4.

	Layout original		Para generación de layout	
	Compacto	Personas con discapacidad	Compacto	Personas con discapacidad
Demanda	56	2	56	2
Ángulo	90	90	90	90
Tipo de módulo	W4	W4	W4	W4
SW (ft)	8.2021	17.7165 (con rampa)	8	17.7165 (con rampa)
PW (ft)	8.2021	17.7165	8	17.7165
# módulos horizontales	2		No factible	
Espacio disponible (ft)	77.09974		77.09974	
Z			114.33333	

Figura 3.26: Comparación de datos de entrada para el estacionamiento 4

$$y = 16.4842' \cos 90^\circ = 0' \tag{3.37}$$

$$PW = \frac{8'}{\sin 90^\circ} = 8' \tag{3.38}$$

$$Número\ de\ cajones\ por\ módulo = \frac{147.9659' - 0'}{8'} * 2 = 36.9914 \approx 36 \tag{3.39}$$

$$Número\ de\ módulos\ horizontales\ para\ cajones\ compactos = \frac{58}{36} = 1.6111 \approx 2 \tag{3.40}$$

$$Z = 2 * 57.1666' = 114.3333' \tag{3.41}$$

En este caso, realizar el layout no es factible (114.3333 no es menor o igual a 77.09974).

3.2.5. Estacionamiento 5

Se tomaron medidas del estacionamiento de plaza Tamarindo. Las medidas reales se muestran en la Figura 3.27.

Ubicación: Carretera, Carmen - Puerto Real 4, Belisario Domínguez, 24150 Cd del Carmen, Camp.

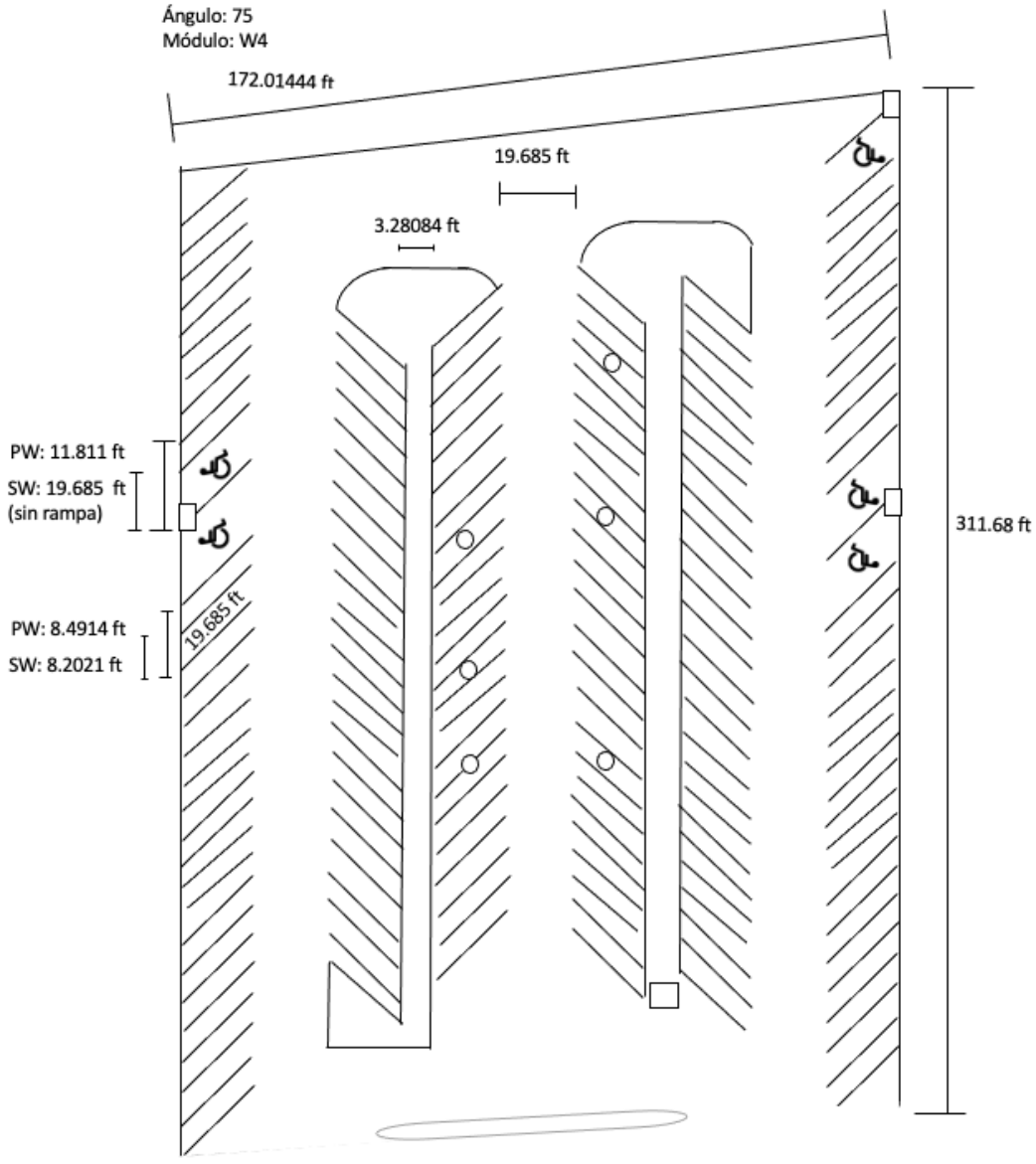


Figura 3.27: Croquis original para estacionamiento 5.

El algoritmo desarrolla la alternativa de layout que se muestra en la Figura 3.28. Los datos que se agregaron son los siguientes.

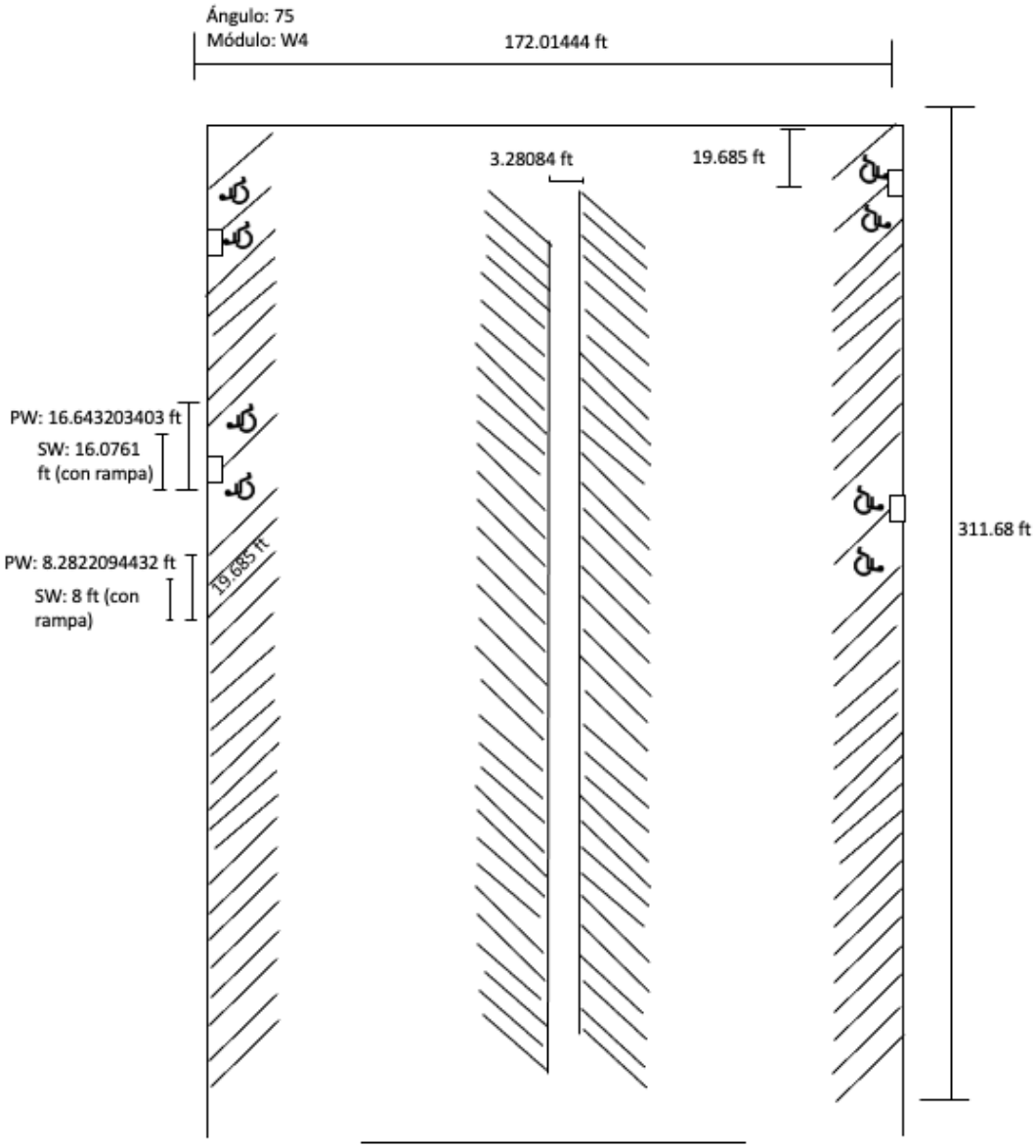


Figura 3.28: Croquis generado en estacionamiento 5.

	Layout original		Para generación de layout	
	Compacto	Personas con discapacidad	Compacto	Personas con discapacidad
Demanda	142	5	142	5
Ángulo	75	75	75	75
Tipo de módulo	W4	W4	W4	W4
SW (ft)	8.2021	11.811 (sin rampa)	8	16.0761 (con ramoa)
PW (ft)	8.4914	12.2276	8	16.6432034
# módulos horizontales	3		2	
Espacio disponible (ft)	172.01444		172.01444	
Z			102.16666	

Figura 3.29: Comparación de datos de entrada para el estacionamiento 5

$$y = 19.685' \cos 75^\circ = 5.0948' \tag{3.42}$$

$$PW = \frac{8'}{\sin 75^\circ} = 8.2822' \tag{3.43}$$

$$Número\ de\ cajones\ por\ módulo = \frac{311.68' - 5.0948'}{8.2822'} * 2 = 74.0347 \approx 74 \tag{3.44}$$

$$Número\ de\ módulos\ horizontales\ para\ cajones\ compactos = \frac{147}{74} = 1.9864 \approx 2 \tag{3.45}$$

$$Z = 2 * 51.0833' = 102.1666' \tag{3.46}$$

En este caso, el desarrollo del layout es factible (102.1666 menor o igual a 172.01444).

	Layout original		Layout generado	
	Compactos	Personas con discapacidad	Compactos	Personas con discapacidad
Fila #1	26	3	28	4
Fila #2	25	0	32	0
Fila #3	21	0	32	0
Fila #4	19	0	28	4
Fila #5	24	0		
Fila #6	27	2		
Total de cajones	142	5	120	8
Demanda	142	5	142	5

Figura 3.30: Comparación de cantidad de cajones para el estacionamiento por fila para el estacionamiento 5

$$Fila\ 1 = \frac{311.68' - 5.0948' - 4 * 16.6432 - 0'}{8.2822094432'} = 28.9793 \approx 28 \tag{3.47}$$

$$Fila\ 2 = \frac{311.68' - 5.0948' - 0 - 2 * 19.685'}{8.2822'} = 32.2637 \approx 32 \tag{3.48}$$

$$Fila\ 3 = \frac{311.68' - 5.0948' - 0 - 2 * 19.685'}{8.2822'} = 32.2637 \approx 32 \tag{3.49}$$

$$Fila\ 4 = \frac{311.68' - 5.0948' - 4 * 16.6432 - 0'}{8.2822094432'} = 28.9793 \approx 28 \tag{3.50}$$

	Layout original	Layout generado
Desventajas	Hay seis árboles ocupando los cajones de estacionamiento.	No cumple con la forma del terreno del estacionamiento. Tampoco, cubre la demanda de automóviles compactos.
Ventajas	Se alinea al terreno del estacionamiento. Cubre la demanda.	Hay mayor número de espacios para automóviles de personas con discapacidad. Se aprovecha de mejor manera el espacio en las filas 2 y 3 del estacionamiento. El espacio de circulación es mayor entre una fila y otra.

Figura 3.31: Comparación de desventajas y ventajas de los layouts para el estacionamiento 5

Parece ser que el croquis generado en el estacionamiento 5 (Figura 3.28) es el menos parecido a su original debido a que el número de filas, módulos, cajones de estacionamiento y terreno no coindicen con el original. Por otro lado, coinciden en ser un estacionamiento vertical.

Capítulo 4

Conclusiones y recomendaciones

La presente investigación tuvo como objetivo desarrollar un algoritmo que facilite la creación de alternativas de layouts. Esta aportación abre la posibilidad a que más personas se involucren en este tema de uso tan contidiano. Este algoritmo aún presenta limitaciones tales como el boceto del layout, porque, aunque da intrucciones escritas y visuales de cómo elaborarlo (Figuras 3.4, 3.5, 3.6 y 3.7), el boceto lo tenemos que dibujar nosotros mismos. Por lo tanto, una recomendación es complementar el código con las instrucciones suficientes para que el programa también dibuje el layout.

En conclusión, el objetivo se cumplió. Una vez concluida la sección 5, se recomienda revisar si las normas americanas cumplen con las de la zona interesada para desarrollar el layout. De no ser así, considerar ajustar las medidas de la tabla de Tompkins et al. (2010). O bien, primero, se pueden cambiar por la normas de la zona. Esto quiere decir que hay dos alternativas para ajustar los parámetros que pide el código. La primera, usar las tablas de Tompkins et al. (2010) y meter los datos al código para, posteriormente, obtener el layout. Lo siguiente es revisar si las normas que se usan en este layout cumplen con las de la zona. Si se cumple, el layout ya está hecho. Segundo, se capturan los datos en el programa conforme a las normas de la zona desde el principio. Así, no habría necesidad de revisar si las de Tompkins et al. (2010) cumplen con las normas de la zona, lo que ahorrraría tiempo.

Apéndice A

Diagrama de flujo de funcionamiento de algoritmo

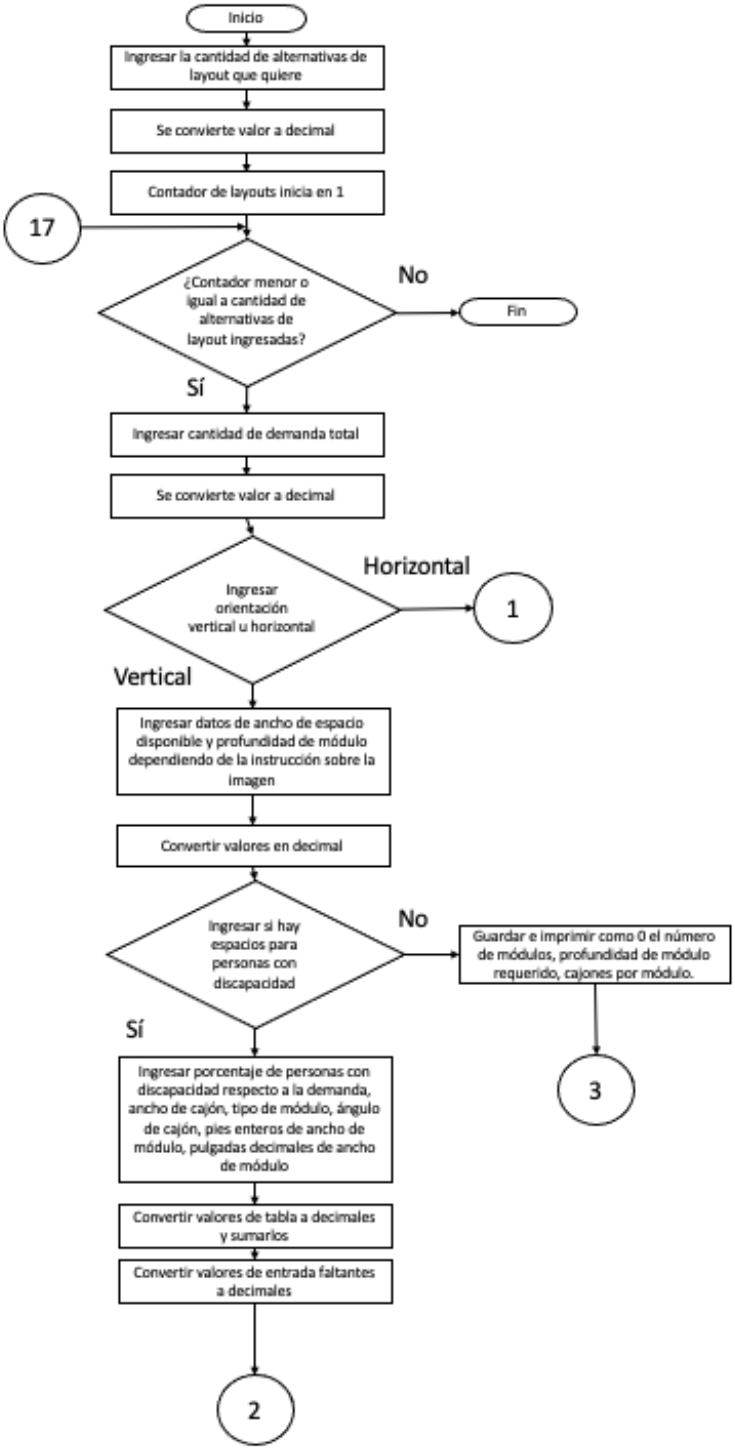


Figura A.1: Diagrama de flujo del algoritmo parte 1

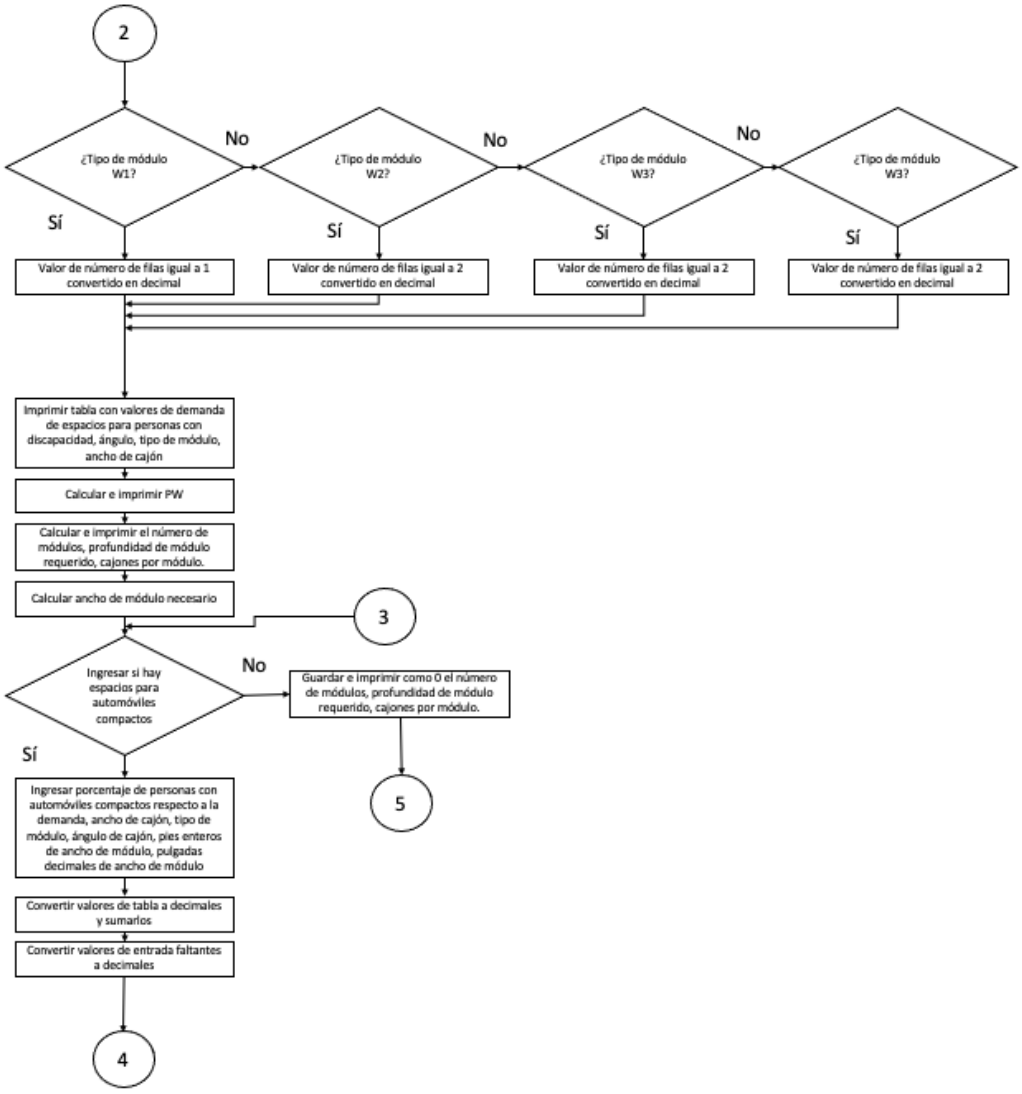


Figura A.2: Diagrama de flujo del algoritmo parte 2

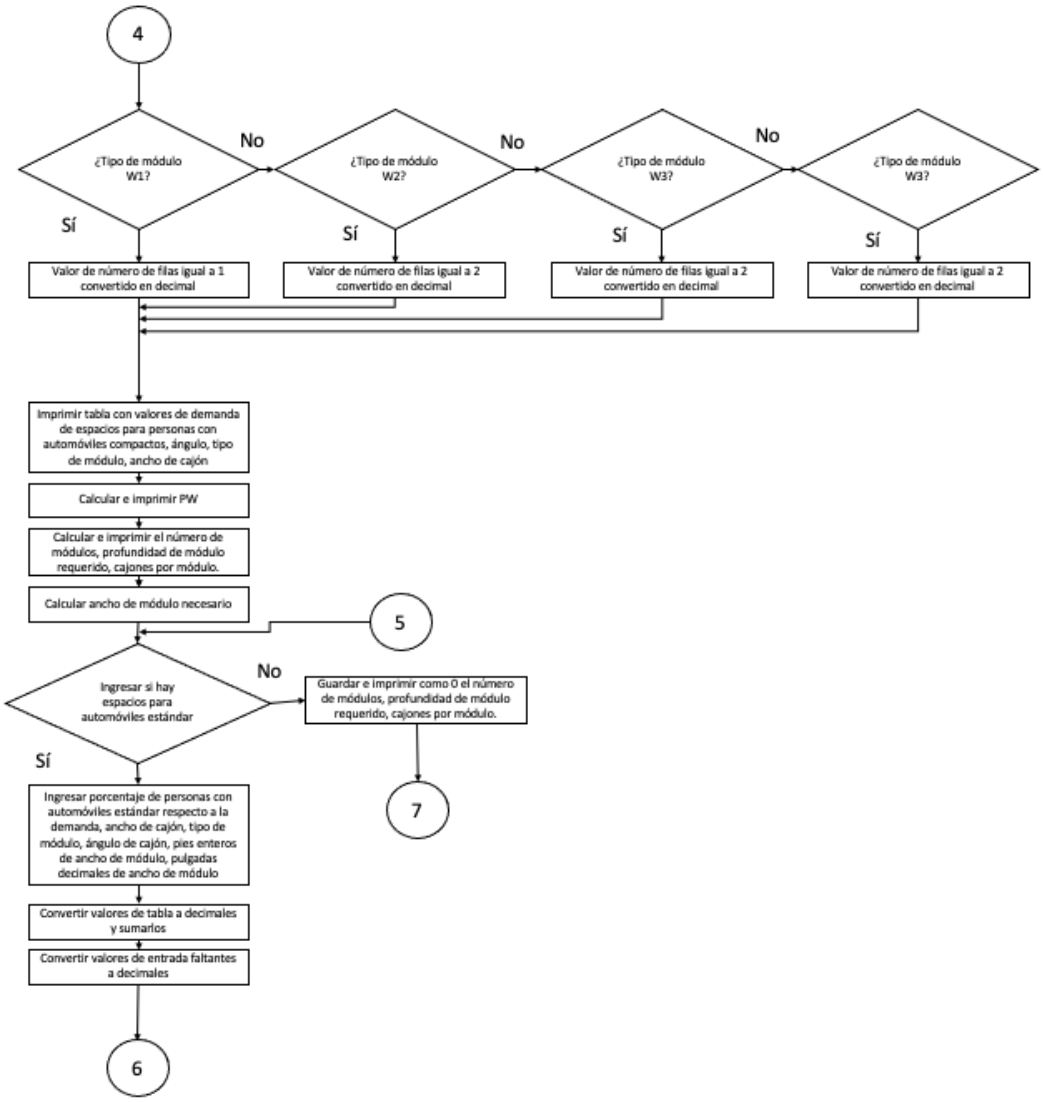


Figura A.3: Diagrama de flujo del algoritmo parte 3

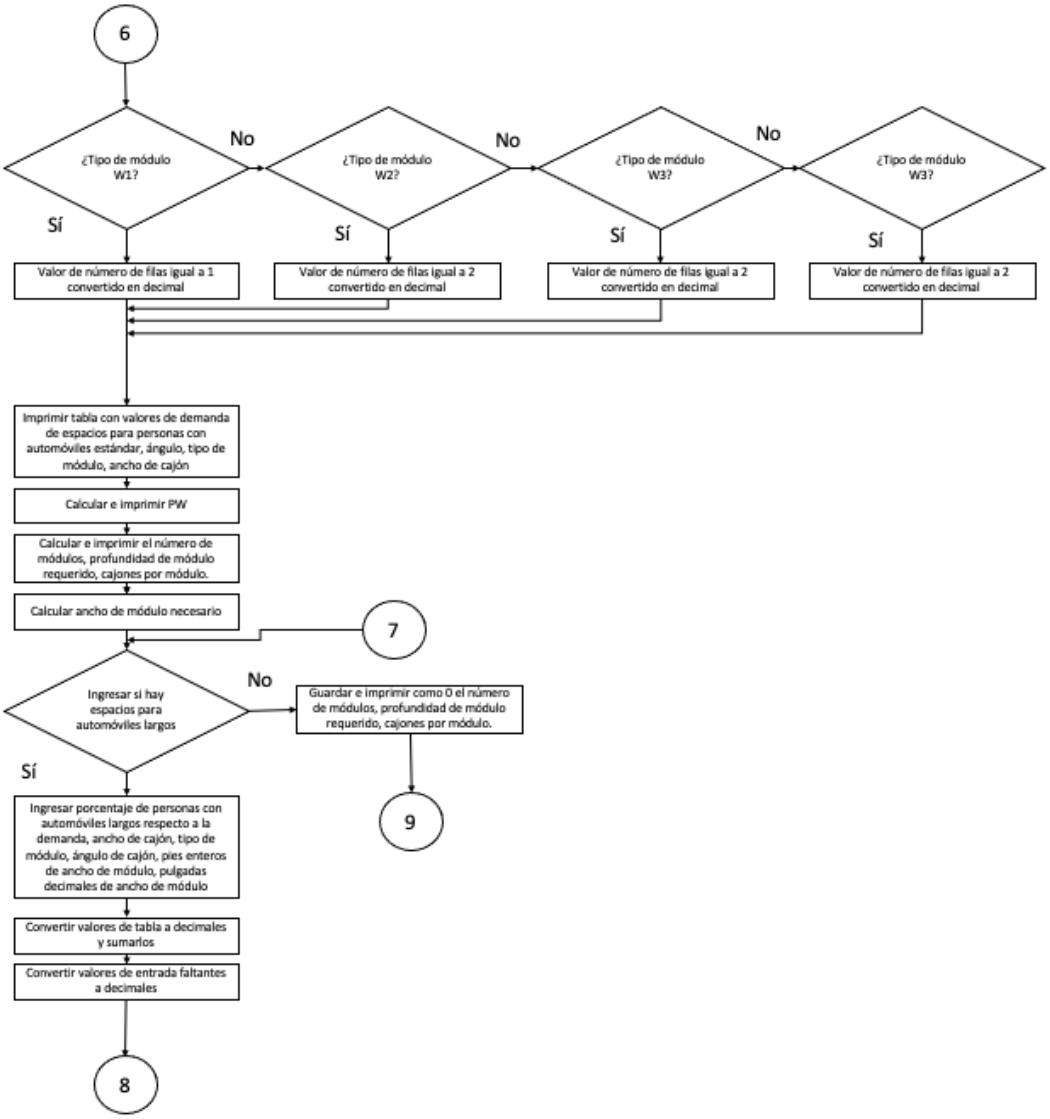


Figura A.4: Diagrama de flujo del algoritmo parte 4

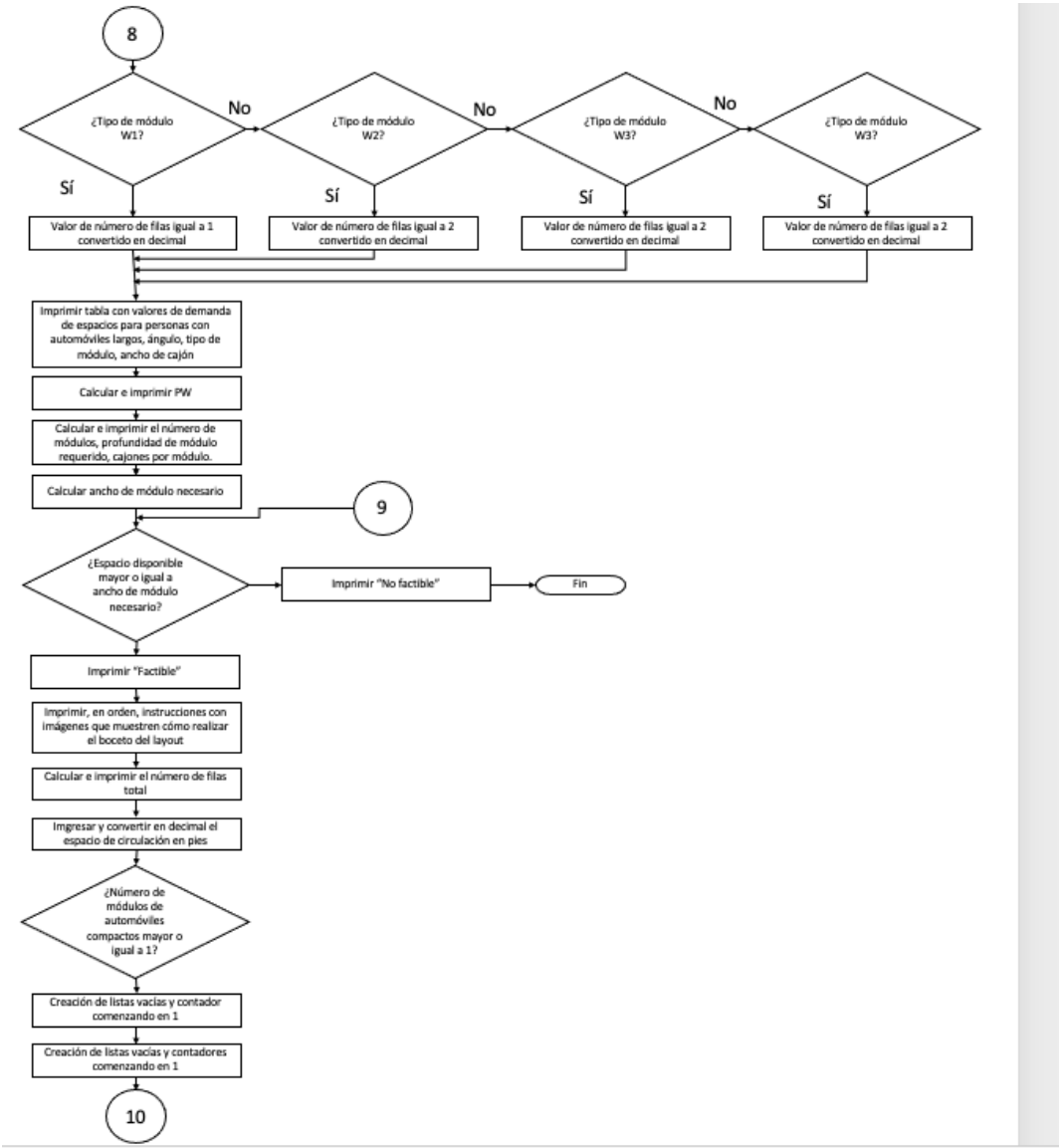


Figura A.5: Diagrama de flujo del algoritmo parte 5

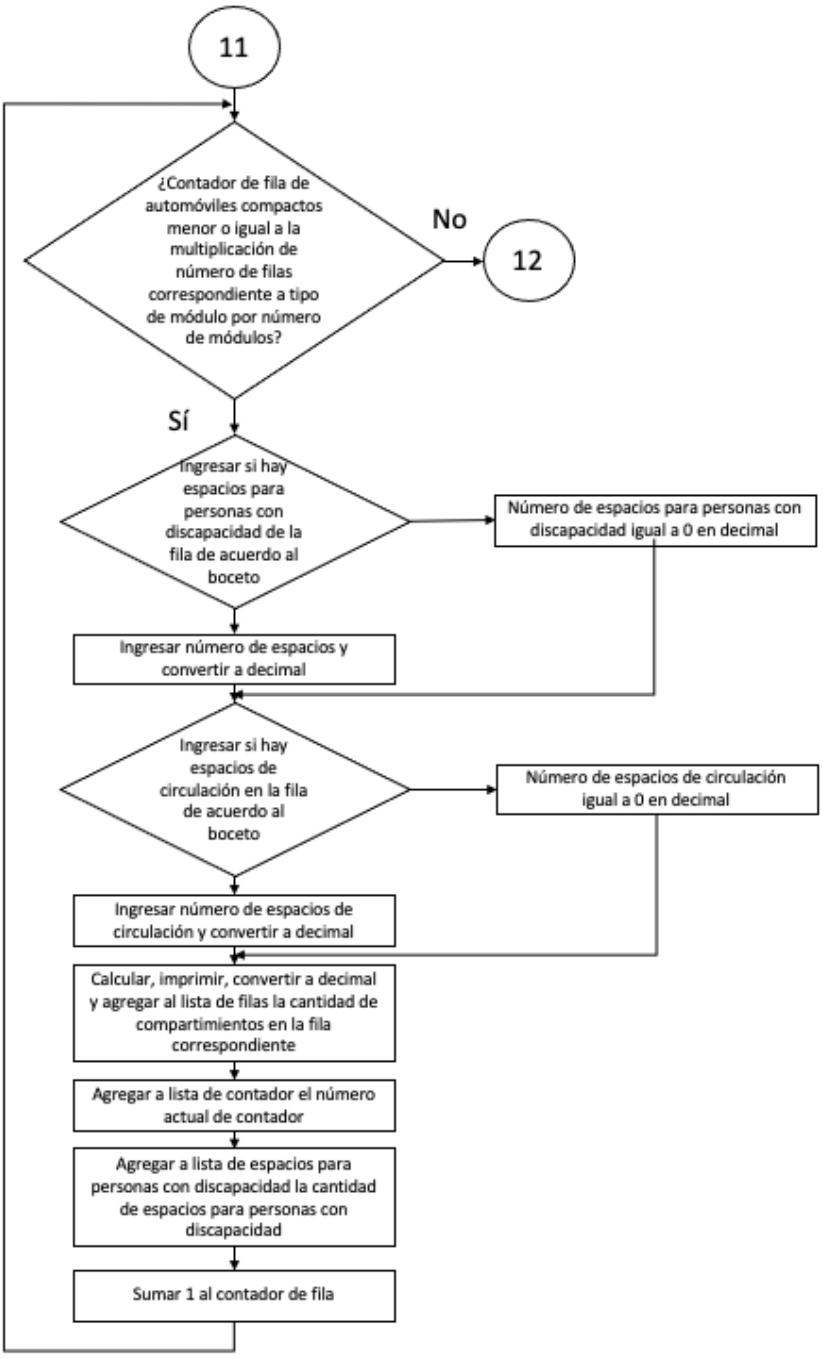


Figura A.6: Diagrama de flujo del algoritmo parte 6

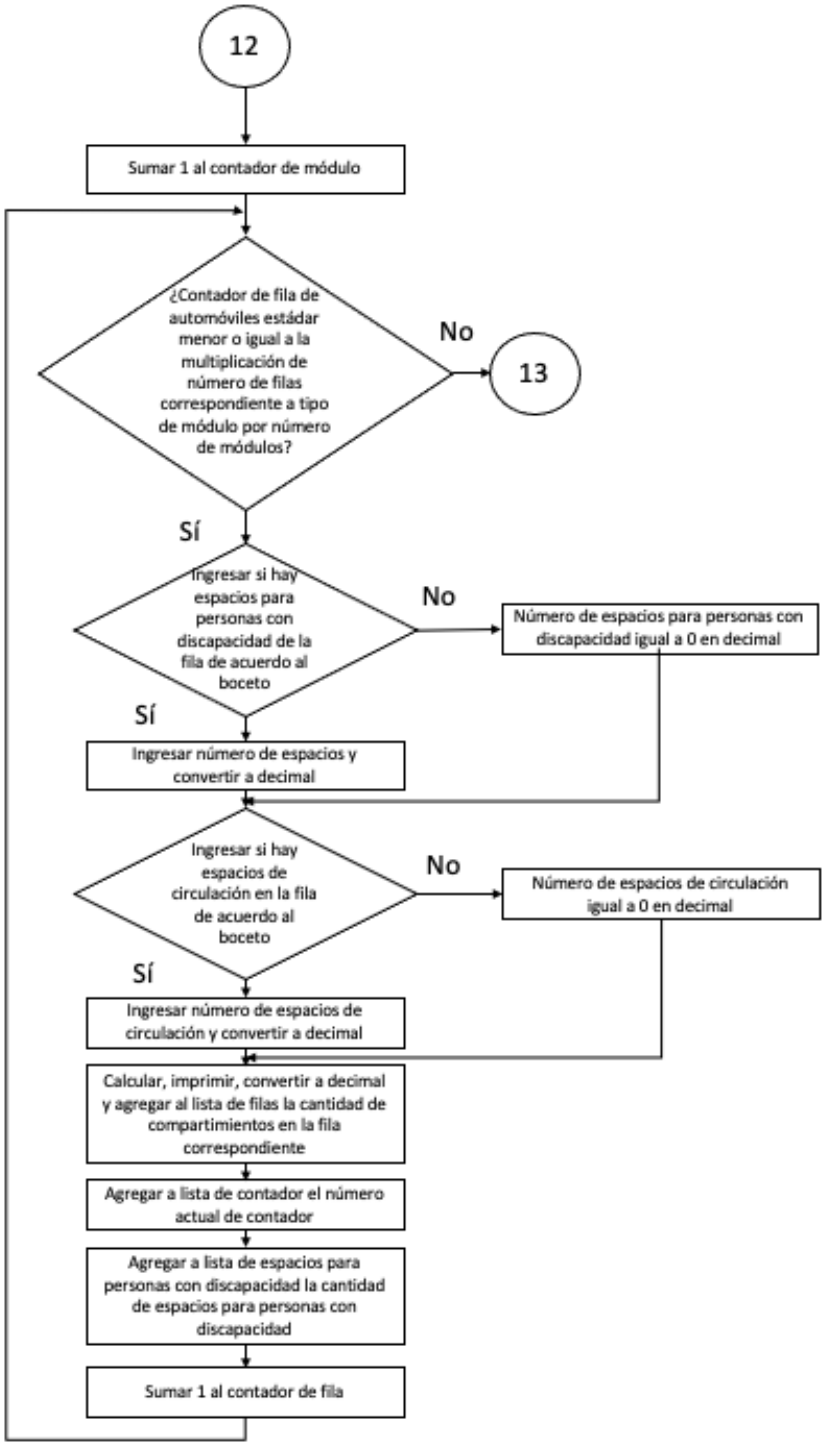


Figura A.7: Diagrama de flujo del algoritmo parte 7

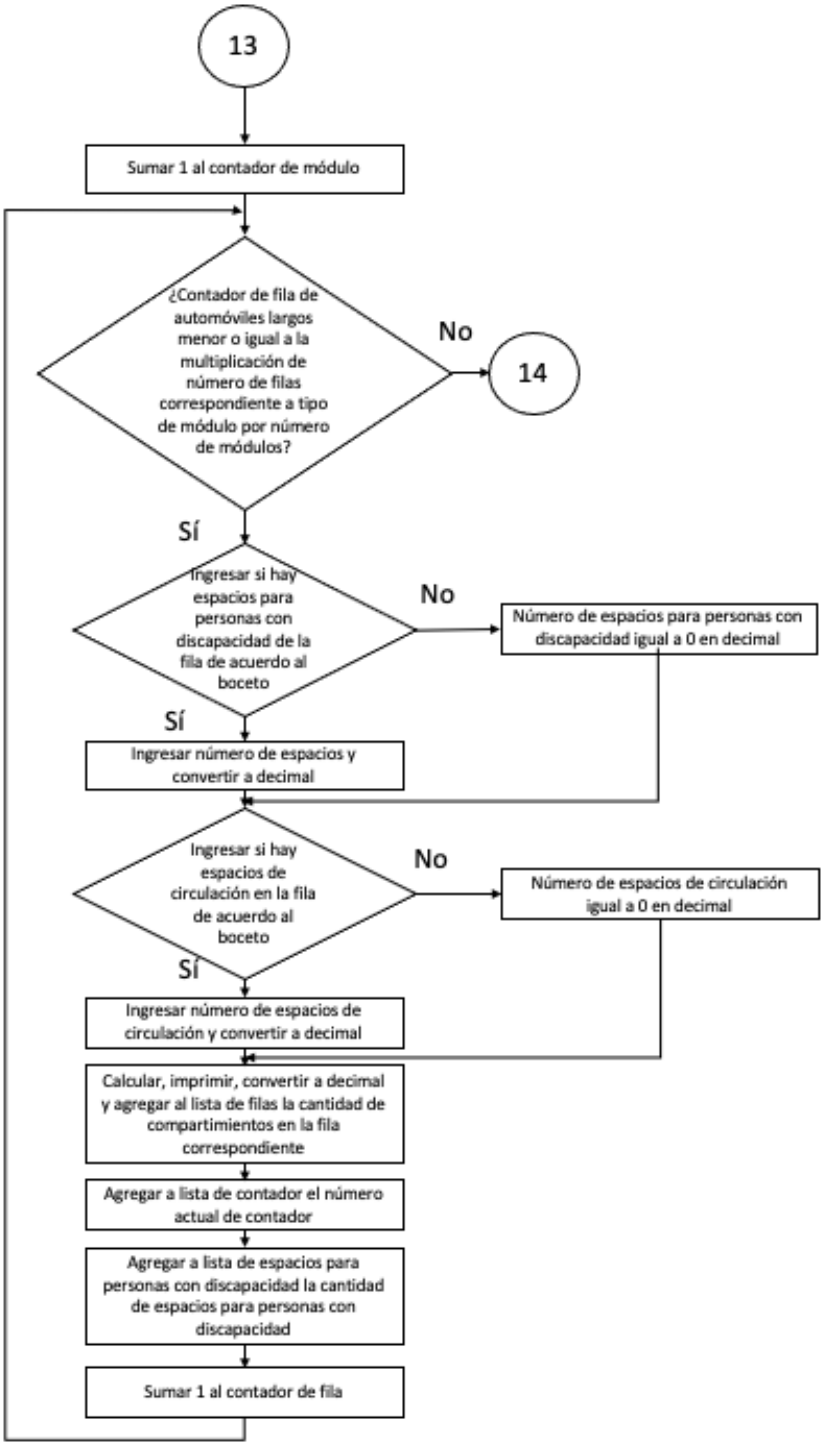


Figura A.8: Diagrama de flujo del algoritmo parte 8

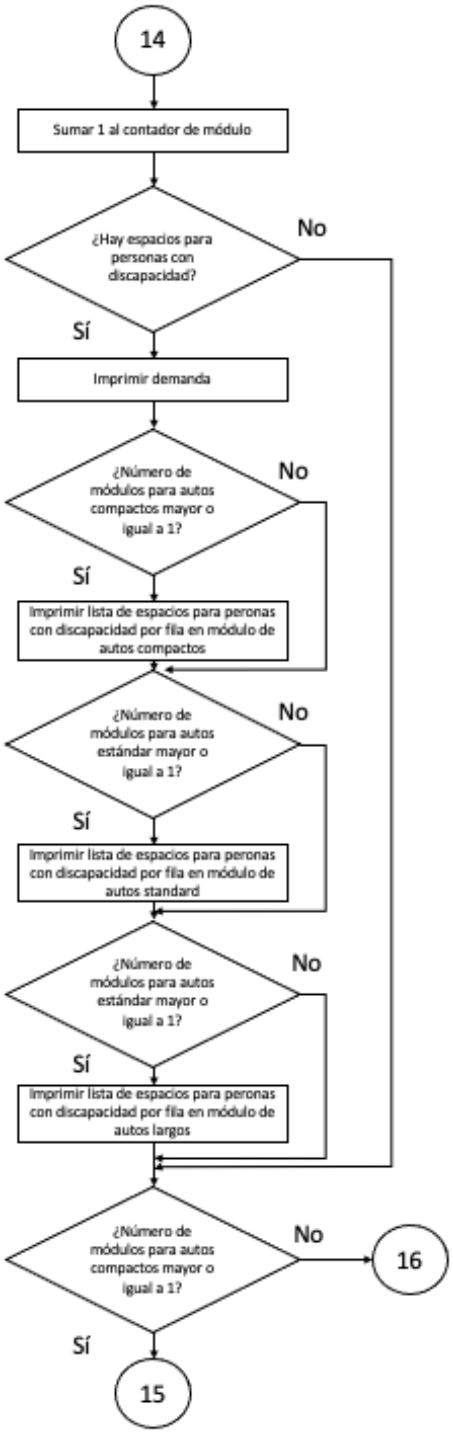


Figura A.9: Diagrama de flujo del algoritmo parte 9

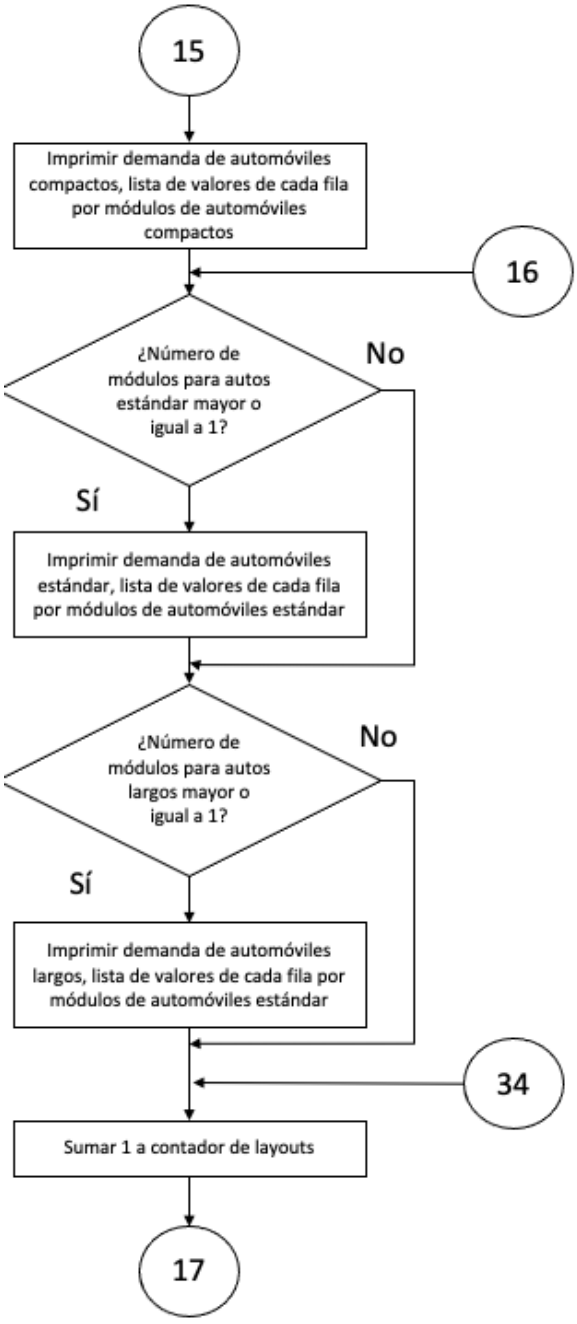


Figura A.10: Diagrama de flujo del algoritmo parte 11

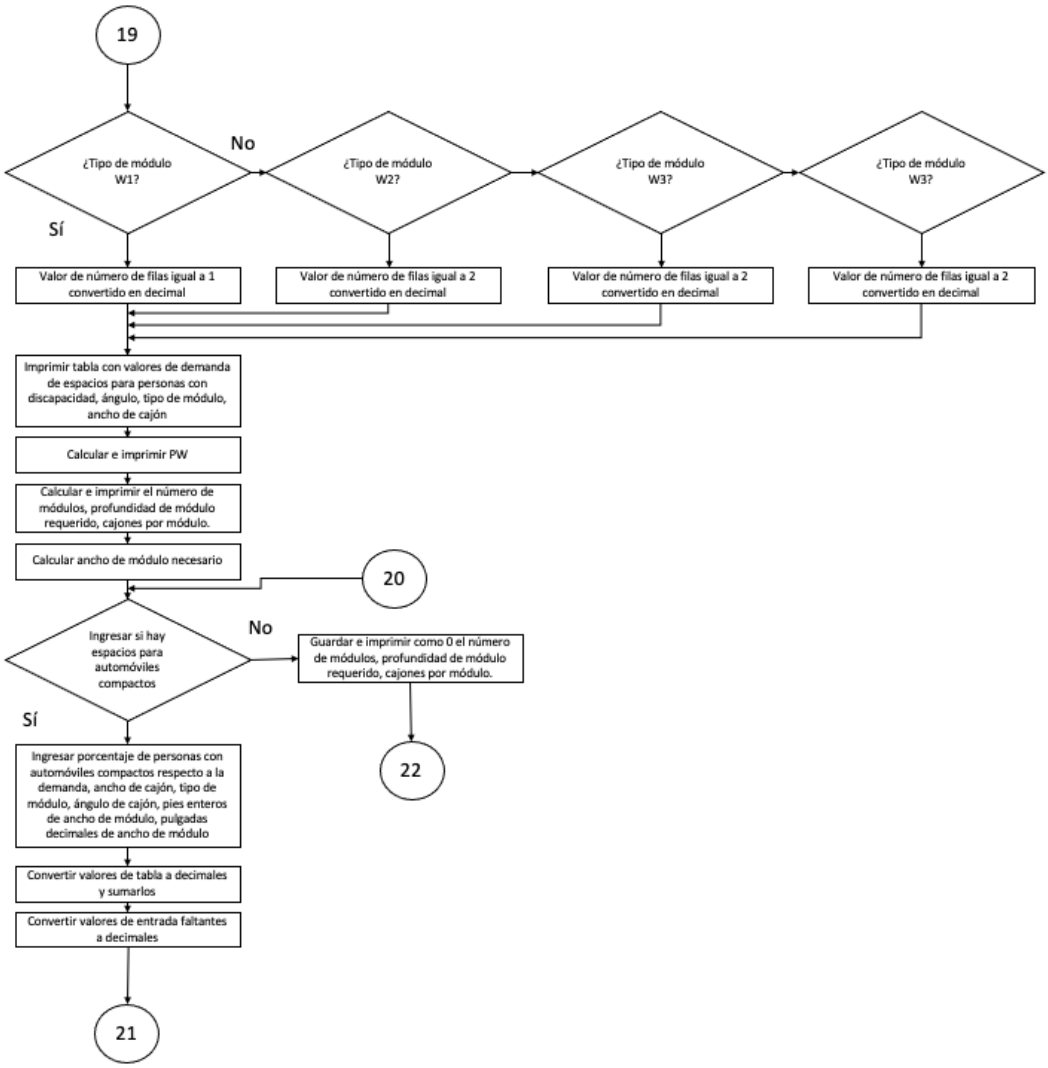


Figura A.11: Diagrama de flujo del algoritmo parte 12

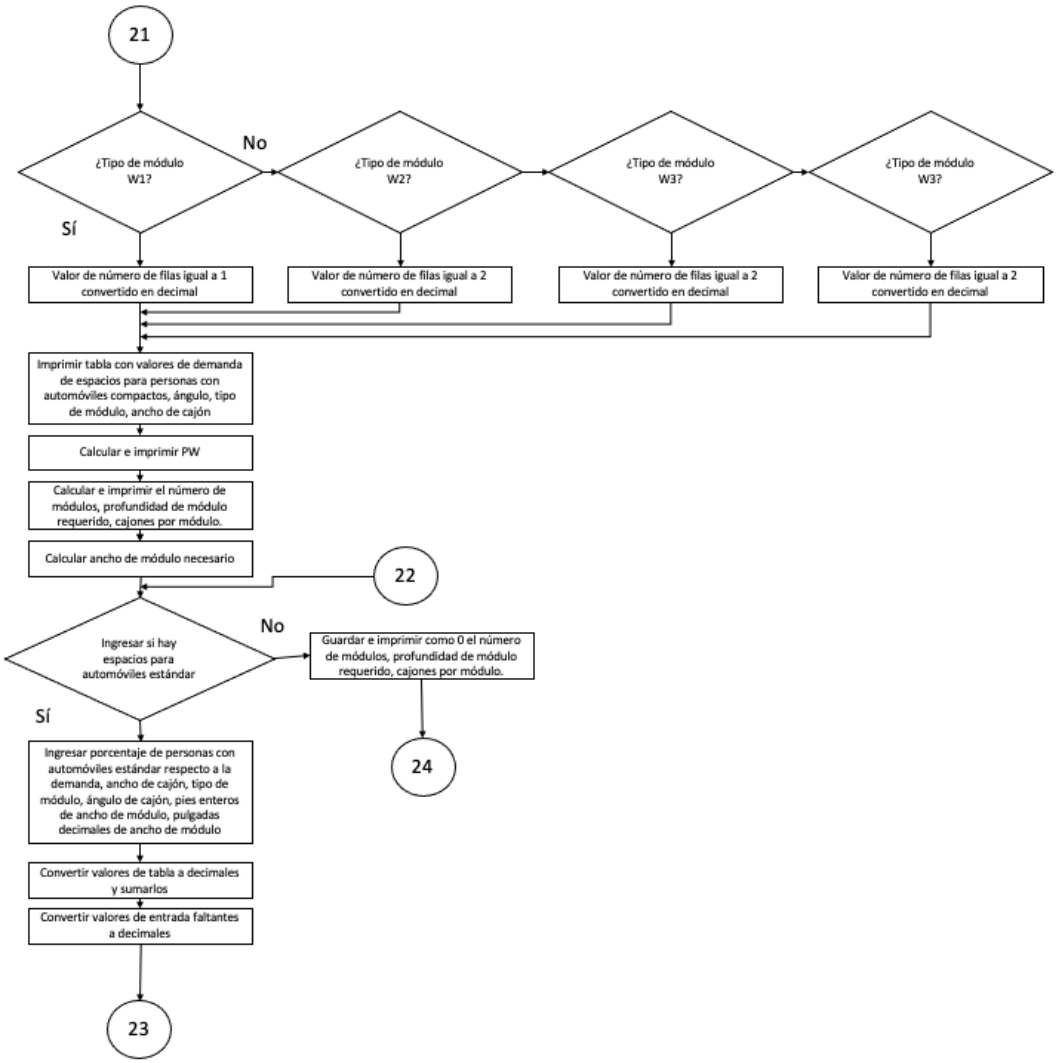


Figura A.12: Diagrama de flujo del algoritmo parte 13

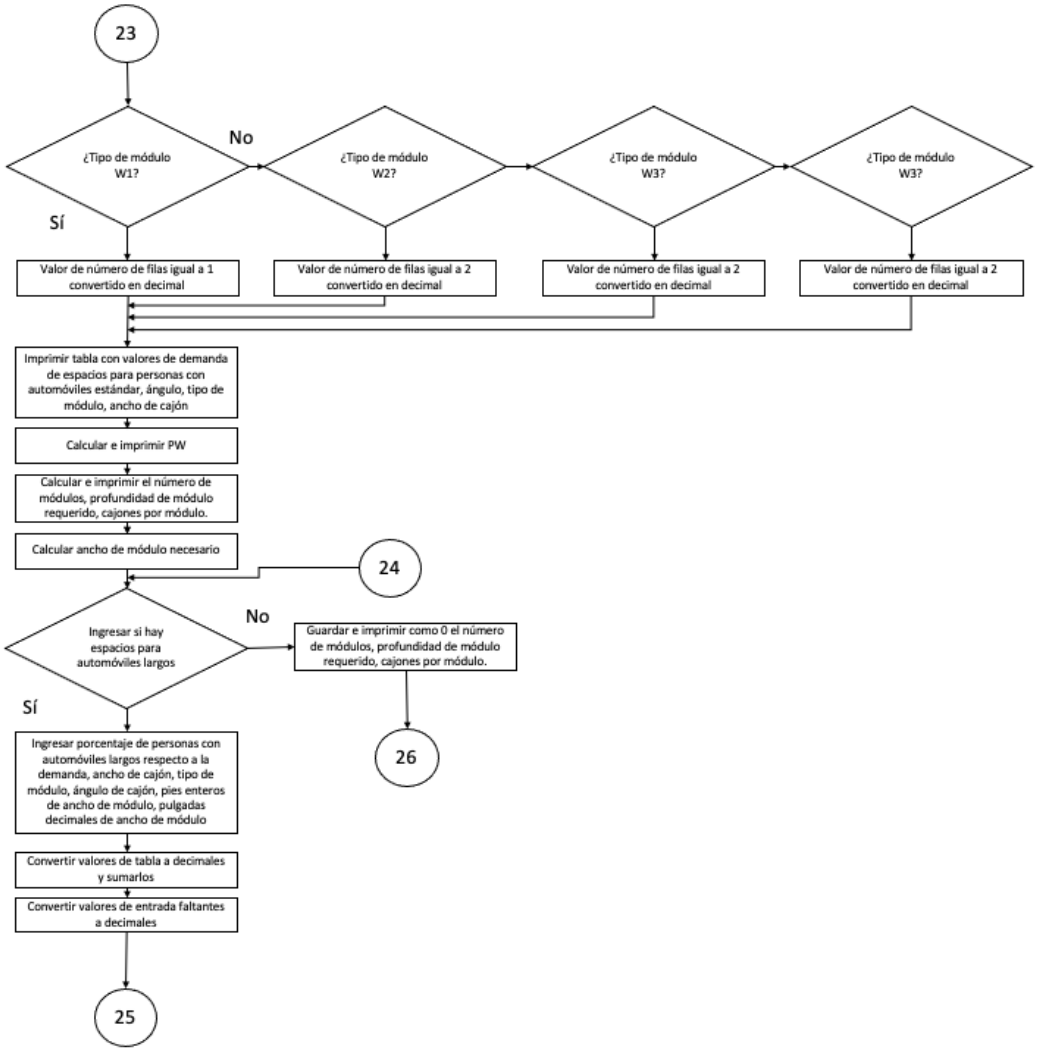


Figura A.13: Diagrama de flujo del algoritmo parte 14

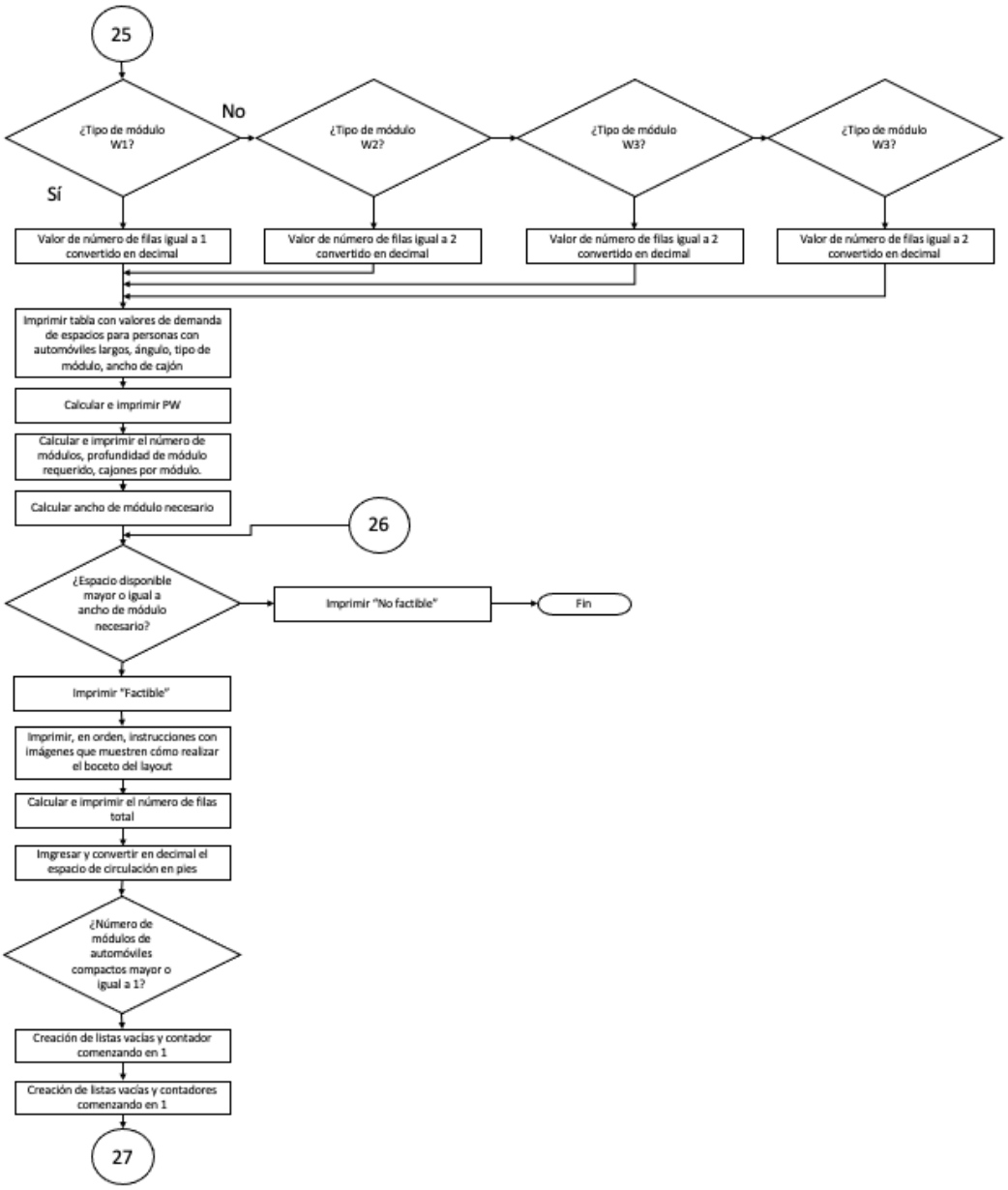


Figura A.14: Diagrama de flujo del algoritmo parte 15

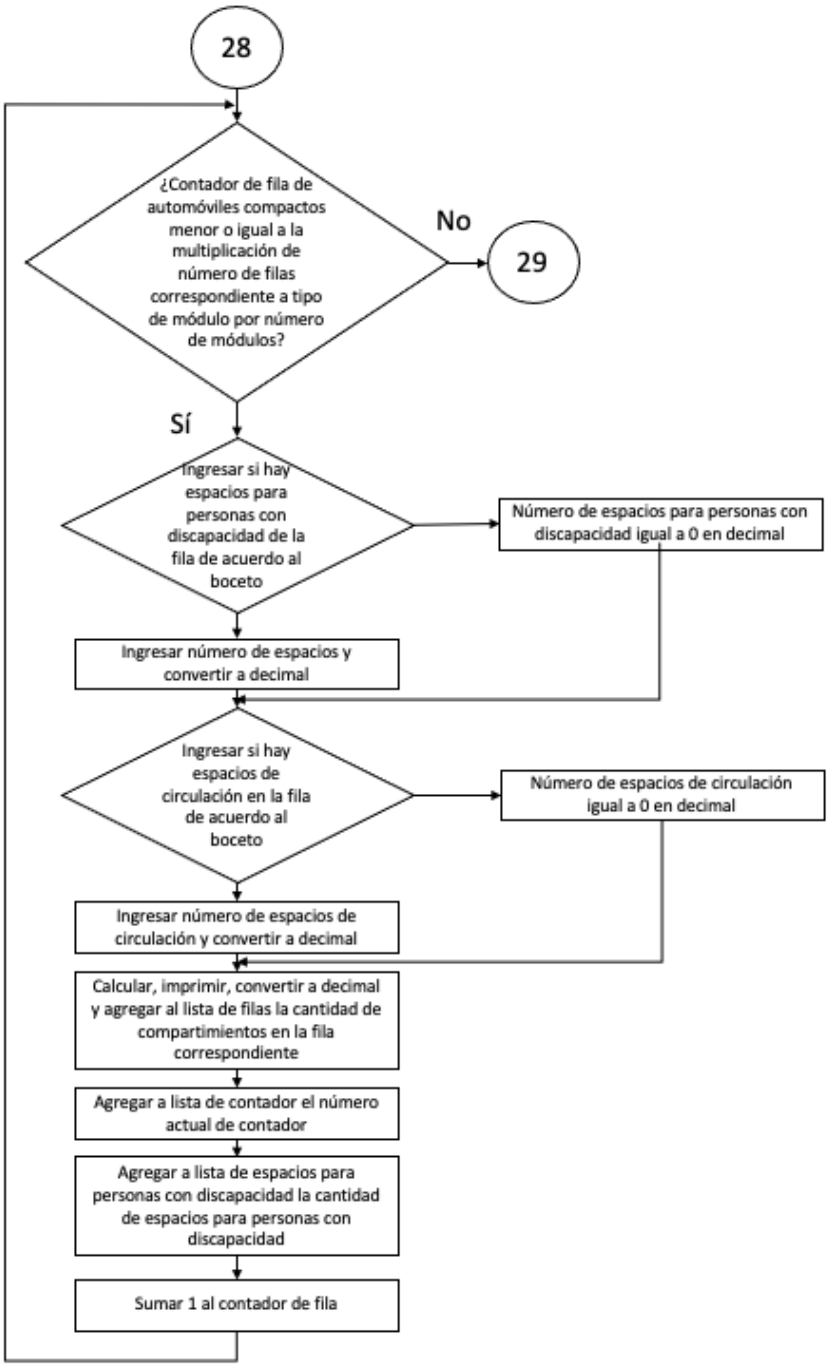


Figura A.15: Diagrama de flujo del algoritmo parte 16

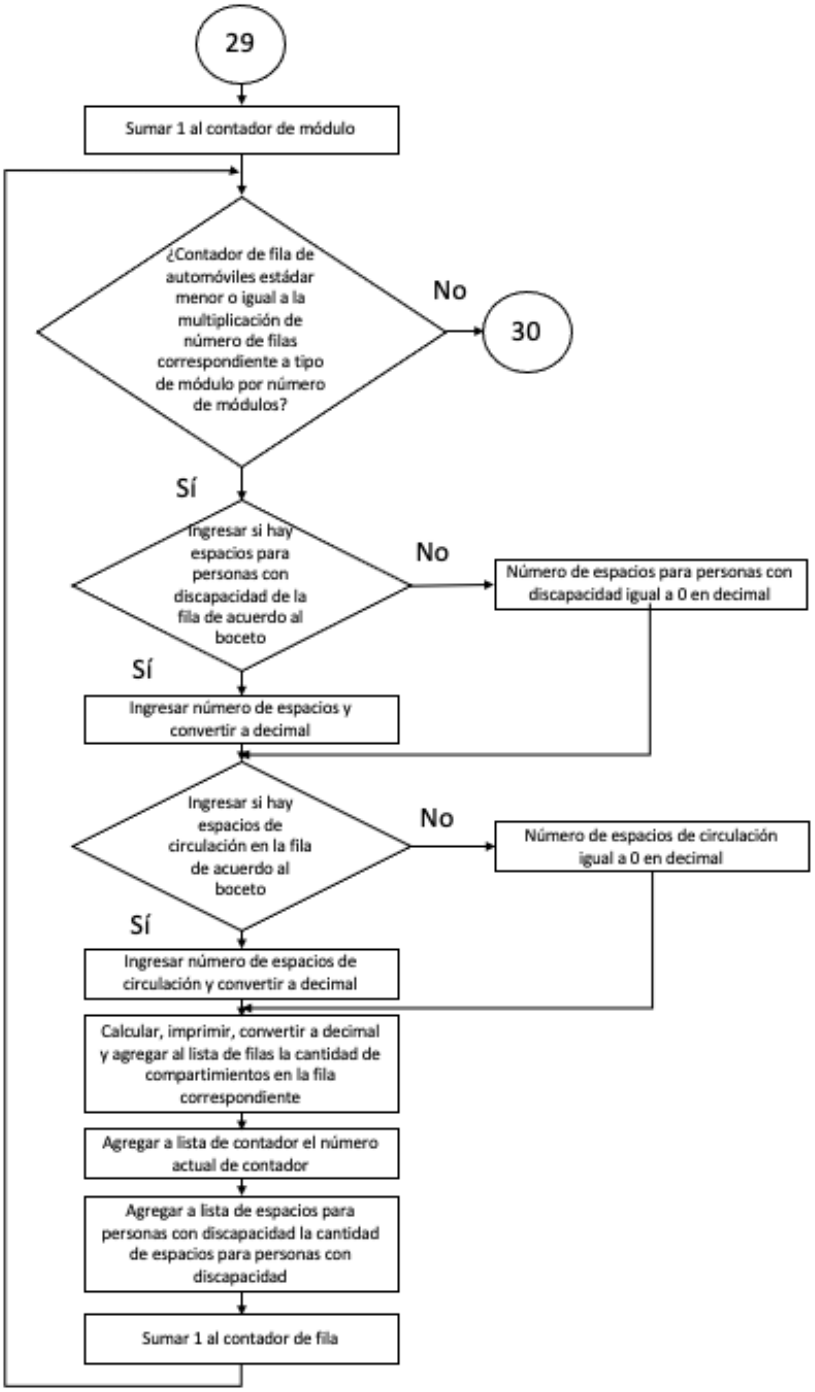


Figura A.16: Diagrama de flujo del algoritmo parte 17

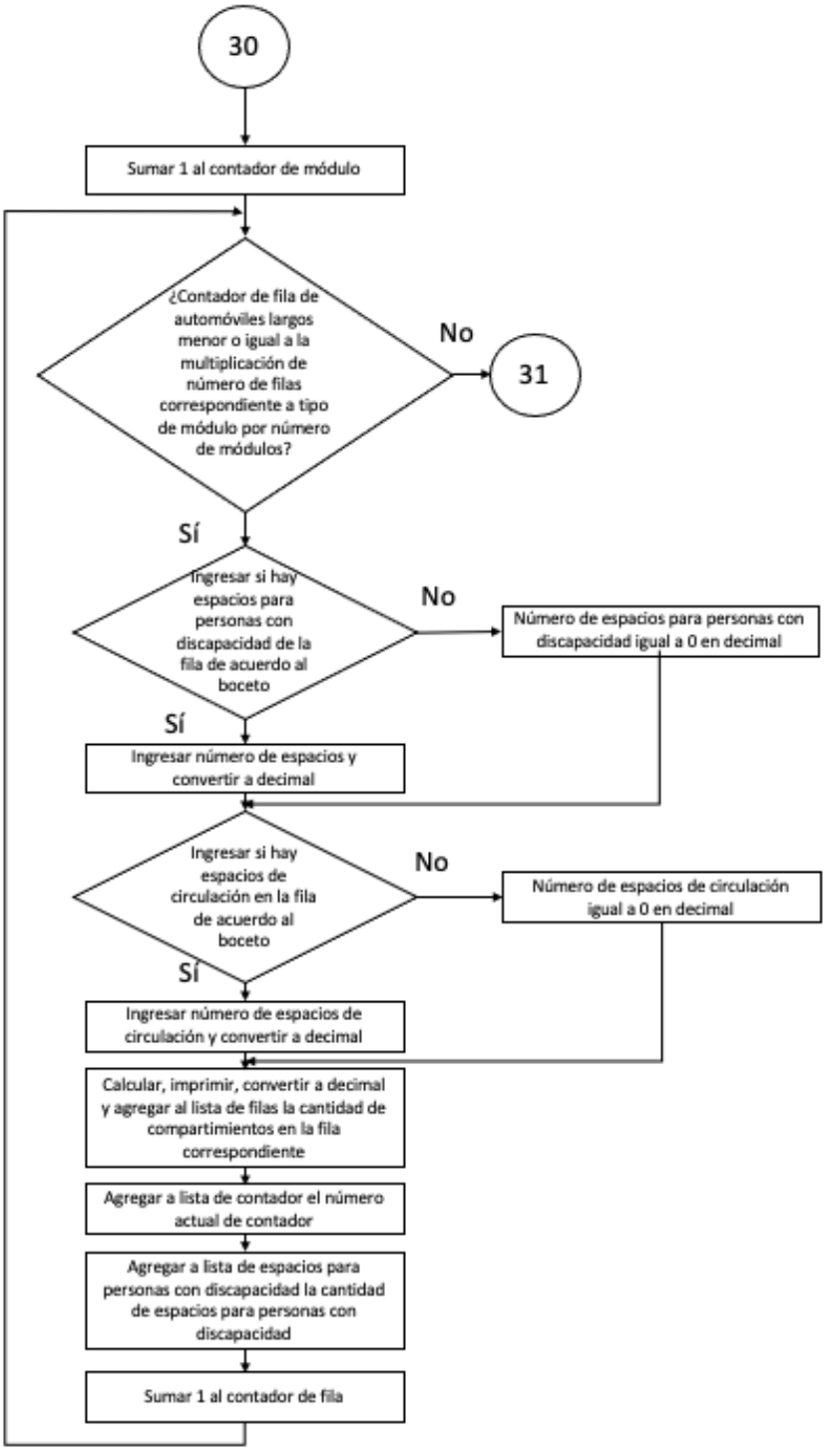


Figura A.17: Diagrama de flujo del algoritmo parte 18

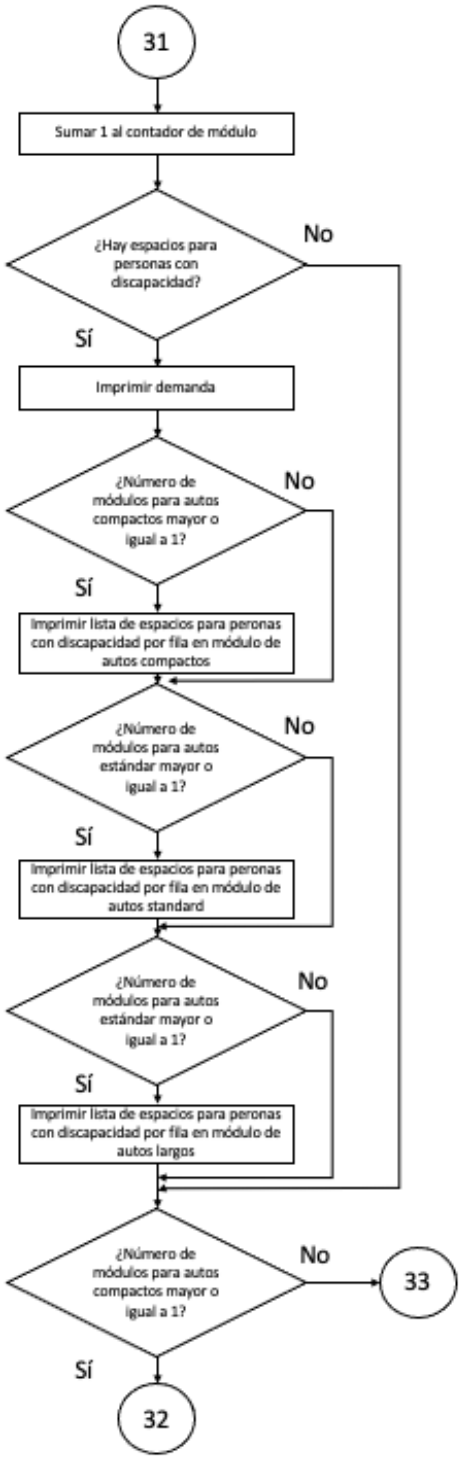


Figura A.18: Diagrama de flujo del algoritmo parte 19

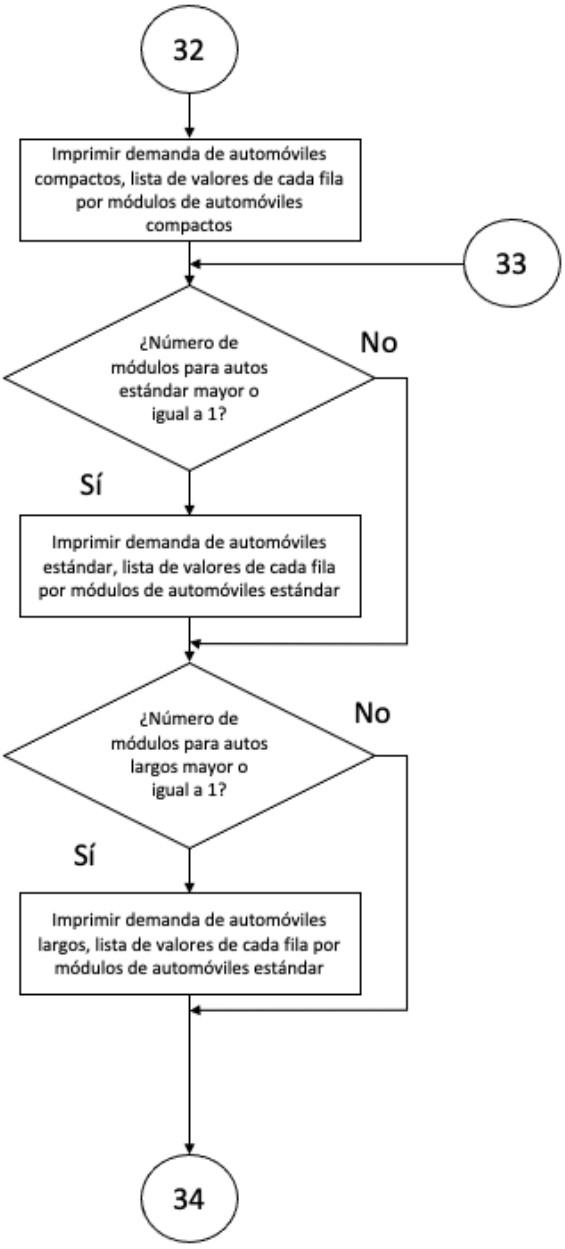


Figura A.19: Diagrama de flujo del algoritmo parte 20

Apéndice B

Algoritmo en python

```
#programa para parking lot

sketch = input('How many layouts do you want? ')
sketch = float(sketch)

count_s = 1
while count_s <= sketch:



   #especificar entre parntesis el dato que se pide, el tipo de
      dato que es
   demand = input('What is the total parking demand? (nmero de
      cajones) ')
   demand = float(demand)
   space = input('Available parking lot space orientation? (
      vertical/horizontal)? ')
   if space == 'vertical':
      from PIL import Image
      print('Watch the image and see how to add the values ')
      im_vertical_ee = Image.open('verticalparking.png')
      im_vertical_ee.show()
      space_vertical = input('Available parking lot space (vertical
         (feet)): ')
      space_vertical = float(space_vertical)
      md = input('Available parking lot space (module depth (feet))
         : ')
```

```
22      md = float(md)
23
24      hc = input("Handicapped space? (yes/no) ")
25      if hc == 'yes':
26          #en esta parte tambin se puede crear una funcin def para
                simplicar
27
28          hm_hc = input('How much? (%) ')
29          sw_hc = input('Choose the stall width ((SW)feet) ') #
                preguntar si el sw siempre es de 12 para HC
30          w_hc = input('Module type (W): ')
31          #rows_hc = input('Rows: ')
32          pa_hc = input('Parking angle: () ')
33          sd_hc = input('Stall Depth (SD (feet)): ')
34          mwtabla_hc1 = input('Check the table and add the feet
                corresponding to parameter given: ')
35          mwtabla_hc2 = input("Check the table and add the '' 
                corresponding to parameter given: ")
36          mwtabla_hc1 = float(mwtabla_hc1)
37          mwtabla_hc2 = float(mwtabla_hc2)
38          mwtabla_hc = float(mwtabla_hc1 + (mwtabla_hc2/12))
39
40  #-----------------------------------------------------------
41  #convertimos los inputs en numeros decimales
42
43          hm_hc = float(hm_hc)
44          sw_hc = float(sw_hc)
45          #rows_hc = float(rows_hc)
```

```
46          pa_hc = float(pa_hc)
47          sd_hc = float(sd_hc)
48
49      #numero de filas por W
50          if w_hc == 'W1':
51              mw_hc = float(1)
52
53          elif w_hc == 'W2':
54              mw_hc = float(2)
55
56          elif w_hc == 'W3':
57              mw_hc = float(2)
58
59          elif w_hc == 'W4':
60              mw_hc = float(2)
61      #-----------------------------------------------------
62          print('Table. Parking layout alternative
                ------------------------------------------')
63          import math
64          print('Parking angle: ', pa_hc)
65          demand_hc = float(demand * hm_hc)
66          redondear_demand_hc = float(math.ceil(demand_hc))
67          print('Demand: ',redondear_demand_hc) #preguntar si se
                redondea hacia arriba o solo redondear
68          print('Parking angle: ', pa_hc)
69          print('Module type: ',w_hc)
70          print('SW: ',sw_hc)
71          #import numpy
```

```
72          #radiandes_pa_hc = ((pa_hc * numpy.pi) / 180)
73          radianes_pa_hc = float(math.radians(pa_hc))
74          pw_hc = float(sw_hc / math.sin(radianes_pa_hc))
75          print('PW:␣',pw_hc)
76          #para handicapped no se pone numero de modulos ni module
               width
77
78          #stall_d_hc = float(sd_hc * math.cos(radianes_pa_hc)) #es
               la y
79          #x_per_row = (md - stall_d_hc) /pw_hc
80          #x_per_row = math.floor(x_per_row) #creo que esta bien
81          #car_stalls_module = x_per_row * rows_hc #revisar si la
               frmula es as
82          #modules_hc = demand_hc / car_stalls_module
83          #print('Vertical # of modules: ',modules_hc)
84          #falta la z y saber si es factible, eso se hace con el
               module width (la tabla)
85
86       #print('Cars which are to be parked')
87    #-------------------------------------------------------
88    #Number of car stalls per module row (parking spots)
89          y_hc = float(sd_hc * math.cos(radianes_pa_hc))
90          x_hc = math.floor((md - y_hc) / pw_hc)
91          #md_r_hc = pw_hc * x_hc + y_hc
92          #cs_per_module = x_hc * mw_hc
93          #module_hc_2 = demand_hc / cs_per_module
94
95          module_hc_2 = float(0)
```

```
96          md_r_hc = float(0)
97
98          cs_per_module = float(0)
99
100         mwtabla_hc = float(0)
101
102         print('Vertical␣#␣of␣modules:␣',module_hc_2)
103         print('Module␣Depth␣required␣=␣',md_r_hc)
104         #cs_per_module = x_hc * mw_hc
105         print('Handicapped␣car␣stalls␣per␣module␣=␣',cs_per_module
                )
106         print('Module␣Width:␣',mwtabla_hc)
107
108     elif hc == 'no':
109
110         module_hc_2 = float(0)
111         md_r_hc = float(0)
112         print('Module␣Depth␣required␣=␣',md_r_hc)
113         cs_per_module = float(0)
114         print('Handicapped␣car␣stalls␣per␣module␣=␣',cs_per_module
                )
115         mwtabla_hc = float(0)
116         print('Module␣Width:␣',mwtabla_hc)
117         mw_hc = float(0)
118         pw_hc = 0
119
120 #-----------------------------------------------------
121 #comenzamos con los autos compactos
```

```
122
123         c = input("Compact? (yes/no) ")
124         if c == 'yes':
125             hm_c = input('How much? ')
126             sw_c = input('Choose the stall width: ') #preguntar si el
                    sw siempre es de 8'' para Compact cars
127             w_c = input('Module type (W): ')
128             #rows_c = input('Rows: ')
129             pa_c = input('Parking angle: ')
130             sd_c = input('Stall Depth (SD): ')
131             mwtabla_c1 = input('Check the table and add the feet
                    corresponding to parameter given: ')
132             mwtabla_c2 = input("Check the table and add the inches
                    corresponding to parameter given: ")
133             mwtabla_c1 = float(mwtabla_c1)
134             mwtabla_c2 = float(mwtabla_c2)
135             mwtabla_c = float(mwtabla_c1 + (mwtabla_c2/12))
136
137     #-----------------------------------------------------
138     #convertimos los inputs en numeros decimales
139
140             hm_c = float(hm_c)
141             sw_c = float(sw_c)
142             #rows_c = float(rows_c)
143             pa_c = float(pa_c)
144             sd_c = float(sd_c)
145
146
```

```
147   #numero de filas por W
148         if w_c == 'W1':
149            mw_c = float(1)
150
151         elif w_c == 'W2':
152            mw_c = float(2)
153
154         elif w_c == 'W3':
155            mw_c = float(2)
156
157         elif w_c == 'W4':
158            mw_c = float(2)
159
160   #-----------------------------------------------------
161
162         print('Table. Parking layout alternative
               ------------------------------------------')
163         import math
164         #print('Parking angle: ', pa_c)
165         demand_c = float(demand * hm_c)
166         redondear_demand_c = float(math.ceil(demand_c))
167         print('Demand (units): ',redondear_demand_c) #preguntar si
               se redondea hacia arriba o solo redondear
168         print('Parking angle (): ', pa_c)
169         print('Module type (W): ',w_c)
170         print('SW (feet): ',sw_c)
171         #import numpy
172         #radiandes_pa_hc = ((pa_hc * numpy.pi) / 180)
```

```
173         radianes_pa_c = float(math.radians(pa_c))
174         pw_c = float(sw_c / math.sin(radianes_pa_c))
175         print('PW (feet): ',pw_c)
176         #para handicapped no se pone numero de modulos ni module
               width
177         #stall_d_c = float(sd_c * math.cos(radianes_pa_c))
178         #x_per_row_c = (md / pw_c) + stall_d_c
179         #entero_x_per_row_c = int(x_per_row_c)
180         #if x_per_row_c >= entero_x_per_row_c:
181         # x_per_row_c = entero_x_per_row_c
182         #car_stalls_module_c = x_per_row_c * rows_c
183         #modules_c = demand_c / car_stalls_module_c
184         #print('Vertical # of modules: ',modules_c)
185         #falta la z y saber si es factible
186   #-----------------------------------------------------------
187   #Number of car stalls per module row (parking spots)
188         y_c = float(sd_c * math.cos(radianes_pa_c))
189         x_c = float(math.floor((md - y_c) / pw_c))
190         md_r_c = float(pw_c * x_c + y_c)
191         cs_per_module_c = float(x_c * mw_c)
192         module_c_2 = (math.ceil(demand_c / cs_per_module_c))
193         module_c_2 = float(module_c_2)
194         print('y= ', y_c)
195         print('x= ', x_c)
196         print('Vertical # modules: ',module_c_2)
197         print('Module Depth required = ',md_r_c)
198         print('Compact car stalls per module = ',cs_per_module_c)
199         print('Module Width (feet): ',mwtabla_c)
```

```
200
201         elif c == 'no':
202
203             module_c_2 = float(0)
204             md_r_c = float(0)
205             print('Module Depth required = ',md_r_c)
206             cs_per_module_c = float(0)
207             print('Handicapped car stalls per module = ',
                    cs_per_module_c)
208             mwtabla_c = float(0)
209             print('Module Width (feet): ',mwtabla_c)
210             mw_c = float(0)
211
212
213
214     #-----------------------------------------------------------
215
216     #autos standard
217         s = input("Standard? (yes/no) ")
218
219         if s == 'yes':
220             hm_s = input('How much? (%) ')
221             sw_s = input('Choose the stall width (SW (feet)): ')
222             w_s = input('Module type (W): ')
223             #rows_s = input('Rows: ')
224             pa_s = input('Parking angle (): ')
225             sd_s = input('Stall Depth (SD (feet)): ')
```

```
226         mwtabla_s1 = input('Check␣the␣table␣and␣add␣the␣feet␣
               corresponding␣to␣parameter␣given:␣')
227         mwtabla_s2 = input("Check␣the␣table␣and␣add␣the␣inches␣
               corresponding␣to␣parameter␣given:␣")
228         mwtabla_s1 = float(mwtabla_s1)
229         mwtabla_s2 = float(mwtabla_s2)
230         mwtabla_s = float(mwtabla_s1 + (mwtabla_s2/12))
231
232     #-----------------------------------------------------------
233     #convertimos los inputs en numeros decimales
234         hm_s = float(hm_s)
235         sw_s = float(sw_s)
236         #rows_s = float(rows_s)
237         pa_s = float(pa_s)
238         sd_s = float(sd_s)
239
240     #numero de filas por W
241         if w_s == 'W1':
242             mw_s = float(1)
243
244         elif w_s == 'W2':
245             mw_s = float(2)
246
247         elif w_s == 'W3':
248             mw_s = float(2)
249
250         elif w_s == 'W4':
251             mw_s = float(2)
```

```
252
253     #-------------------------------------------------------
254           print('Table. Parking layout alternative
                 ------------------------------------------')
255           import math
256           #print('Parking angle (): ', pa_s)
257           demand_s = float(demand * hm_s)
258           redondear_demand_s = float(math.ceil(demand_s))
259           print('Demand (units): ',redondear_demand_s) #preguntar si
                  se redondea hacia arriba o solo redondear
260           print('Parking angle (): ', pa_s)
261           print('Module type (W): ',w_s)
262           print('SW (feet): ',sw_s)
263           #import numpy
264           #radiandes_pa_hc = ((pa_hc * numpy.pi) / 180)
265           radianes_pa_s = float(math.radians(pa_s))
266           pw_s = float(sw_s / math.sin(radianes_pa_s))
267           print('PW (feet): ',pw_s)
268           #para handicapped no se pone numero de modulos ni module
                 width
269           #stall_d_s = float(sd_s * math.cos(radianes_pa_s))
270           #x_per_row_s = (md / pw_s) + stall_d_s
271           #entero_x_per_row_s = int(x_per_row_s)
272           #if x_per_row_s >= entero_x_per_row_s:
273           # x_per_row_s = entero_x_per_row_s
274           #car_stalls_module_s = x_per_row_s * rows_s
275           #modules_s = demand_s / car_stalls_module_s
276           #print('Vertical # of modules: ',modules_s)
```

```
277         #falta la z y saber si es factible
278     #-------------------------------------------------------
279     #Number of car stalls per module row (parking spots)
280         y_s = float(sd_s * math.cos(radianes_pa_s))
281         x_s = float(math.floor((md - y_s) / pw_s))
282         md_r_s = float(pw_s * x_s + y_s)
283         cs_per_module_s = float(x_s * mw_s)
284         module_s_2 = (math.ceil(demand_s / cs_per_module_s))
285         module_s_2 = float(module_s_2)
286         print('Vertical # of modules: ',module_s_2)
287         print('Module Depth required = ',md_r_s)
288         print('Handicapped car stalls per module = ',
                cs_per_module_s)
289         print('Module Width (feet): ',mwtabla_s)
290
291       elif s == 'no':
292
293         module_s_2 = float(0)
294         md_r_s = float(0)
295         print('Module Depth required = ',md_r_s)
296         cs_per_module_s = float(0)
297         print('Handicapped car stalls per module = ',
                cs_per_module_s)
298         mwtabla_s = float(0)
299         print('Module Width (feet): ',mwtabla_s)
300         mw_s = float(0)
301
302
```

```
303  #-----------------------------------------------------------
304  #Large
305
306     l = input("Large? (yes/no) ")
307
308     if l == 'yes':
309        hm_l = input('How much? ( %) ')
310        sw_l = input('Choose the stall width (SW (feet)): ')
311        w_l = input('Module type (W): ')
312        #rows_l = input('Rows: ')
313        pa_l = input('Parking angle ():')
314        sd_l = input('Stall Depth (SD (feet)): ')
315        mwtabla_l1 = input('Check the table and add the feets
              corresponding to parameter given: ')
316        mwtabla_l2 = input("Check the table and add the inches
              corresponding to parameter given: ")
317        mwtabla_l1 = float(mwtabla_l1)
318        mwtabla_l2 = float(mwtabla_l2)
319        mwtabla_l = float(mwtabla_l1 + (mwtabla_l2/12))
320
321  #-----------------------------------------------------------
322  #convertimos los inputs en numeros decimales
323        hm_l = float(hm_l)
324        sw_l = float(sw_l)
325        #rows_l = float(rows_l)
326        pa_l = float(pa_l)
327        sd_l = float(sd_l)
328
```

```
329   #numero de filas por W
330         if w_l == 'W1':
331            mw_l = float(1)
332
333         elif w_l == 'W2':
334            mw_l = float(2)
335
336         elif w_l == 'W3':
337            mw_l = float(2)
338
339         elif w_l == 'W4':
340            mw_l = float(2)
341
342   #-----------------------------------------------------------
343         print('Table. Parking layout alternative
               ------------------------------------------')
344         import math
345         print('Parking angle ():', pa_l)
346         demand_l = float(demand * hm_l)
347         redondear_demand_l = math.ceil(demand_l)
348         print('Demand: ',redondear_demand_l) #preguntar si se
               redondea hacia arriba o solo redondear
349         print('Parking angle ():', pa_l)
350         print('Module type (W): ',w_l)
351         print('SW (feet): ',sw_l)
352         #import numpy
353         #radiandes_pa_hc = ((pa_hc * numpy.pi) / 180)
354         radianes_pa_l = float(math.radians(pa_l))
```

```
355         pw_1 = float(sw_1 / math.sin(radianes_pa_1))
356         print('PW (feet): ',pw_1)
357         #para handicapped no se pone numero de modulos ni module
               width
358         #stall_d_1 = float(sd_1 * math.cos(radianes_pa_1))
359         #x_per_row_1 = (md / pw_1) + stall_d_1
360         #entero_x_per_row_1 = int(x_per_row_1)
361         #if x_per_row_1 >= entero_x_per_row_1:
362         # x_per_row_1 = entero_x_per_row_1
363         #car_stalls_module_1 = x_per_row_1 * rows_1
364         #modules_1 = demand_1 / car_stalls_module_1
365         #print('Vertical # of modules: ',modules_1)
366         #falta la z y saber si es factible
367
368     #-------------------------------------------------------
369
370     #Number of car stalls per module row (parking spots)
371           y_1 = float(sd_1 * math.cos(radianes_pa_1))
372           x_1 = float(math.floor((md - y_1) / pw_1))
373           md_r_1 = float(pw_1 * x_1 + y_1)
374           cs_per_module_1 = float(x_1 * mw_1)
375           module_1_2 = (math.ceil(demand_1 / cs_per_module_1))
376           module_1_2 = float(module_1_2)
377           print('Vertical # of module: ',module_1_2)
378           print('Module Depth required (feet) = ',md_r_1)
379           print('Handicapped car stalls per module = ',
                 cs_per_module_1)
380           print('Module Width (feet): ',mwtabla_1)
```

```
381
382         elif l == 'no':
383
384             module_l_2 = float(0)
385             md_r_l = float(0)
386             print('Module Depth required (feet) = ',md_r_l)
387             cs_per_module_l = float(0)
388             print('Large car stalls per module = ',cs_per_module_l)
389             mwtabla_l = float(0)
390             print('Module Width (feet): ',mwtabla_l)
391             mw_l = float(0)
392
393
394     #-----------------------------------------------------
395     #Feasible?
396         z = float((module_hc_2 * mwtabla_hc) + (module_c_2 *
                mwtabla_c) + (module_s_2 * mwtabla_s) + (module_l_2 *
                mwtabla_l))
397         print('Z = ',z)
398         print(space_vertical)
399         if space_vertical >= z:
400             print('Factible')
401         else:
402             print('No factible')
403     #-----------------------------------------------------
404     #imagen y sketch parte 1
405
406         if space_vertical >= z:
```

```
407
408         print("Draw␣an␣sketch␣similar␣to␣the␣image␣depending␣on␣#␣
                of␣modules␣of␣each␣type␣of␣car")
409
410         from PIL import Image
411
412         # creating a object
413         im_vertical = Image.open('vertical.png')
414
415         im_vertical.show()
416
417         print('Consider␣external␣communication␣channels␣as␣in␣the␣
                image␣')
418         im_vertical_channels = Image.open('channels.png')
419         im_vertical_channels.show()
420
421         print('Consider␣entrance␣and␣exit␣')
422         im_vertical_ee = Image.open('entrance␣and␣exit.png')
423         im_vertical_ee.show()
424
425         print('Consider␣circulation␣lines␣in␣the␣sketch␣')
426         im_vertical_circulation = Image.open('circulation␣lines.
                png')
427         im_vertical_circulation.show()
428
429         print('Consider␣handicapped␣spaces␣according␣to␣the␣table␣
                and␣the␣area␣you␣want␣to␣add␣them')
```

```
430         im_vertical_hcspaces = Image.open('handicapped␣spaces.png'
                )
431         im_vertical_hcspaces.show()
432
433       #calculos de las filas
434         total_rows = mw_hc * module_hc_2 + mw_c * module_c_2 +
                mw_s * module_s_2 + mw_l * module_l_2 #el de
                handicapped no es necesario, solo lo puse porque en
                este baso todo pero los dems hay que agregarlos y
                sumarlos
435         print('Total␣rows:␣',total_rows)
436         circulation_lines = input('How␣many␣feet␣for␣circulation␣
                lines?␣')
437         circulation_lines = float(circulation_lines)
438
439         #tomar en cuenta poner un if para si autos compactos/
                standard/large vertical # of modules >= 1
440
441
442       #para hc
443       # if module_hc_2 >= 1:
444       # print('For rows in Handicapped cars ')
445       # count_hc = 0
446             #lista_hc = []
447             #lista_hc = append(count_hc)
448             #for num in lista_hc:
449       # while count_hc <= module_hc_2:
```

```
450     # spacehc_hc = input('Are there any spaces for handicapped
            cars? ')
451     # if spacehc_hc == 'yes':
452     # spacehc_h_hc = input('How many? ')
453     # spacehc_h_hc = float(spacehc_h_hc)
454     # elif spacehc_hc == 'no':
455     # spacehc_h_hc = 0
456     # spacehc_h_hc = float(spacehc_h_hc)
457     # spacecl_hc = input('Are there any spaces for cirulation
            lines? ')
458     # if spacecl_hc == 'yes':
459     # spacecl_h_hc = input('How many? ')
460     # spacecl_h_hc = float(spacecl_h_hc)
461     # #spacecl_hi_hc = input('How many inches of each circulation
            line? ')
462     # #spacecl_hi_hc = float(spacecl_hi_hc)
463     # elif spacecl_hc == 'no':
464     # spacecl_h_hc = 0
465     # spacecl_h_hc = float(spacecl_h_hc)
466     #
467     #
468     # row_hc = float((md - (spacehc_h_hc * pw_hc) - (spacecl_h_hc
            * circulation_lines) - y_hc) / pw_hc)
469     # print('Row ',count_hc, '=', row_hc)
470
471     # count_hc = count_hc + 1
472
473     #para compact rows
```

```
474
475         if module_c_2 >= 1:
476             #countmodule_c = 1
477             lista_c = []
478             lista_hc_c = []
479             lista_row_c = []
480             count_c = 1
481
482             count_r_c = 1
483             #lista_hc_c = append(spacehc_h_c)
484             print('For␣rows␣in␣compact␣cars␣')
485             while count_r_c <= (mw_c * module_c_2):
486                 spacehc_c = input('Are␣there␣any␣spaces␣for␣
                        handicapped␣cars?␣')
487                 if spacehc_c == 'yes':
488                     spacehc_h_c = input('How␣many?␣')
489                     spacehc_h_c = float(spacehc_h_c)
490                 elif spacehc_c == 'no':
491                     spacehc_h_c = 0
492                     spacehc_h_c = float(spacehc_h_c)
493                 spacecl_c = input('Are␣there␣any␣spaces␣for␣
                        cirulation␣lines?␣')
494                 if spacecl_c == 'yes':
495                     spacecl_h_c = input('How␣many?␣')
496                     spacecl_h_c = float(spacecl_h_c)
497                           #spacecl_hi_hc = input('How many inches of
                                each circulation line? ')
498                           #spacecl_hi_hc = float(spacecl_hi_hc)
```

```
499                elif spacecl_c == 'no':
500                  spacecl_h_c = 0
501                  spacecl_h_c = float(spacecl_h_c)
502
503
504                row_c = float((md - (spacehc_h_c * pw_hc) - (
                       spacecl_h_c * circulation_lines) - y_c) / pw_c)
505                print('Row ',count_r_c, '=', row_c)
506                lista_row_c.append(row_c)
507                lista_c.append(count_r_c)
508                lista_hc_c.append(spacehc_h_c)
509
510                count_r_c = count_r_c + 1
511             count_c = count_c + 1
512
513
514       #para standard
515
516          if module_s_2 >= 1:
517             lista_s = []
518             lista_hc_s = []
519             lista_row_s = []
520
521             count_s = 1
522
523             print('For rows in standard cars ')
524                #count_s = 1
525                #lista_s = []
```

```
526                 #lista_s.append(count_s)
527
528                 #lista_hc_s = []
529                 #lista_row_1 = []
530                 #lista_hc = []
531                 #lista_hc = append(count_hc)
532                 #for num in lista_hc:
533             count_r_s = 1
534             #lista_hc_s = append(spacehc_h_s)
535             while count_r_s <= (mw_s * module_s_2):
536                 #lista_hc_s = append(spacehc_h_s)
537                 spacehc_s = input('Are there any spaces for
                        handicapped cars? ')
538                 if spacehc_s == 'yes':
539                     spacehc_h_s = input('How many? ')
540                     spacehc_h_s = float(spacehc_h_s)
541                 elif spacehc_s == 'no':
542                     spacehc_h_s = 0
543                     spacehc_h_s = float(spacehc_h_s)
544                 spacecl_s = input('Are there any spaces for
                        cirulation lines? ')
545                 if spacecl_s == 'yes':
546                     spacecl_h_s = input('How many? ')
547                     spacecl_h_s = float(spacecl_h_s)
548                         #spacecl_hi_hc = input('How many inches of
                                each circulation line? ')
549                         #spacecl_hi_hc = float(spacecl_hi_hc)
550                 elif spacecl_s == 'no':
```

```
551                 spacecl_h_s = 0
552                 spacecl_h_s = float(spacecl_h_s)
553
554
555             row_s = float((md - (spacehc_h_s * pw_hc) - (
                    spacecl_h_s * circulation_lines) - y_s) / pw_s)
556             print('Row ',count_r_s, '=', row_s)
557             lista_row_s.append(row_s)
558             lista_s.append(count_s)
559             lista_hc_s.append(spacehc_h_s)
560             count_r_s = count_r_s + 1
561          count_s = count_s + 1
562
563       #para large rows
564       countmodule_l = 1
565       if module_l_2 >= 1:
566          lista_l = []
567          lista_hc_l = []
568          lista_row_l = []
569          count_l = 1
570
571          print('For rows in large cars ')
572
573             #lista_l = []
574             #lista_l.append(count_l)
575
576             #lista_hc_l = []
577             #lista_row_l = []
```

```
578                 #lista_hc = []
579             #lista_hc = append(count_hc)
580             count_r_l = 1
581                 #for num in lista_hc:
582             while count_r_l <= (mw_l * module_l_2):
583                     #lista_hc_l = append(spacehc_h_l)
584                 spacehc_l = input('Are there any spaces for
                        handicapped cars? ')
585                 if spacehc_l == 'yes':
586                     spacehc_h_l = input('How many? ')
587                     spacehc_h_l = float(spacehc_h_l)
588                 elif spacehc_l == 'no':
589                     spacehc_h_l = 0
590                     spacehc_h_l = float(spacehc_h_l)
591                 spacecl_l = input('Are there any spaces for
                        cirulation lines? ')
592                 if spacecl_l == 'yes':
593                     spacecl_h_l = input('How many? ')
594                     spacecl_h_l = float(spacecl_h_l)
595                         #spacecl_hi_hc = input('How many inches of
                                each circulation line? ')
596                         #spacecl_hi_hc = float(spacecl_hi_hc)
597                 elif spacecl_l == 'no':
598                     spacecl_h_l = 0
599                     spacecl_h_l = float(spacecl_h_l)
600
601
```

```
602                 row_l = float((md - (spacehc_h_l * pw_hc) - (
                        spacecl_h_l * circulation_lines) - y_l) / pw_l)
603                 print('Row ',count_r_l, '=', row_l)
604                 lista_row_l.append(row_l)
605                 lista_l.append(count_l)
606                 lista_hc_l.append(spacehc_h_l)
607
608                 count_r_l = count_r_l + 1
609             count_l = count_l + 1
610
611         print("
                Resumen_____________________________________________________
                ")
612
613         if hc == 'yes':
614             print('Demand per handicapped cars: ',demand_hc)
615             #for row_hc_c2 in lista_c:
616             if module_c_2 >= 1:
617                 print('Compact car Row ',lista_hc_c,': ',lista_hc_c)
618             #for row_hc_s2 in lista_s:
619             if module_s_2 >= 1:
620                 print('Standard car Row ',lista_hc_s,': ',lista_hc_s
                        )
621             #for row_hc_l2 in lista_l:
622             if module_l_2 >= 1:
623                 print('Standard car Row ',lista_hc_l,': ',lista_hc_l
                        )
624
```

```
625         if module_c_2 >= 1:
626             print('Demand␣per␣compact␣cars:␣',demand_c)
627             #for row_c_c2 in lista_c:
628             print (lista_c)
629             print('Compact␣cars␣per␣Row␣')
630             print('Row:␣',lista_c)
631             print('Stalls:␣',lista_row_c)
632 
633         if module_s_2 >= 1:
634             print('Demand␣per␣standard␣cars:␣',demand_s)
635             #for row_s_s2 in lista_s:
636             print('Standard␣cars␣per␣Row␣')
637             print('Row:␣',lista_s)
638             print('Stalls:␣',lista_row_s)
639 
640         if module_l_2 >= 1:
641             print('Demand␣per␣large␣cars:␣',demand_l)
642             #for row_c_c2 in lista_l:
643             print('Large␣car␣per␣Row␣')
644             print('Row:␣',lista_l)
645             print('Stalls:␣',lista_row_l)
646 
647     elif space == 'horizontal':
648         from PIL import Image
649         print('Watch␣the␣image␣and␣see␣how␣to␣add␣the␣values␣')
650         im_vertical_ee = Image.open('horizontalparking.png')
651         im_vertical_ee.show()
```

```
652         space_vertical = input('Available␣parking␣lot␣space␣(
               horizontal␣(feet)):␣')
653         space_vertical = float(space_vertical)
654         md = input('Available␣parking␣lot␣space␣(module␣depth␣(feet))
               :␣')
655         md = float(md)
656
657         hc = input("Handicapped␣space?␣(yes/no)␣")
658         if hc == 'yes':
659             #en esta parte tambin se puede crear una funcin def para
                   simplicar
660
661             hm_hc = input('How␣much?␣(%)␣')
662             sw_hc = input('Choose␣the␣stall␣width:␣(feet)␣') #
                   preguntar si el sw siempre es de 12 para HC
663             w_hc = input('Module␣type␣(W):␣')
664             #rows_hc = input('Rows: ')
665             pa_hc = input('Parking␣angle:␣()␣')
666             sd_hc = input('Stall␣Depth␣(SD␣(feet)):␣')
667             mwtabla_hc1 = input('Check␣the␣table␣and␣add␣the␣feet␣
                   corresponding␣to␣parameter␣given:␣')
668             mwtabla_hc2 = input("Check␣the␣table␣and␣add␣the␣''␣
                   corresponding␣to␣parameter␣given:␣")
669             mwtabla_hc1 = float(mwtabla_hc1)
670             mwtabla_hc2 = float(mwtabla_hc2)
671             mwtabla_hc = float(mwtabla_hc1 + (mwtabla_hc2/12))
672
673     #-------------------------------------------------
```

```
674    #convertimos los inputs en numeros decimales
675
676          hm_hc = float(hm_hc)
677          sw_hc = float(sw_hc)
678          #rows_hc = float(rows_hc)
679          pa_hc = float(pa_hc)
680          sd_hc = float(sd_hc)
681
682    #numero de filas por W
683          if w_hc == 'W1':
684             mw_hc = float(1)
685
686          elif w_hc == 'W2':
687             mw_hc = float(2)
688
689          elif w_hc == 'W3':
690             mw_hc = float(2)
691
692          elif w_hc == 'W4':
693             mw_hc = float(2)
694    #-------------------------------------------------
695          print('Table. Parking layout alternative
                ----------------------------------------------')
696          import math
697          print('Parking angle: ', pa_hc)
698          demand_hc = float(demand * hm_hc)
699          redondear_demand_hc = float(math.ceil(demand_hc))
```

```
700         print('Demand: ',redondear_demand_hc) #preguntar si se
               redondea hacia arriba o solo redondear
701         print('Parking angle: ', pa_hc)
702         print('Module type: ',w_hc)
703         print('SW: ',sw_hc)
704         #import numpy
705         #radiandes_pa_hc = ((pa_hc * numpy.pi) / 180)
706         radianes_pa_hc = float(math.radians(pa_hc))
707         pw_hc = float(sw_hc / math.sin(radianes_pa_hc))
708         print('PW: ',pw_hc)
709         #para handicapped no se pone numero de modulos ni module
               width
710
711         #stall_d_hc = float(sd_hc * math.cos(radianes_pa_hc)) #es
               la y
712         #x_per_row = (md - stall_d_hc) /pw_hc
713         #x_per_row = math.floor(x_per_row) #creo que esta bien
714         #car_stalls_module = x_per_row * rows_hc #revisar si la
               frmula es as
715         #modules_hc = demand_hc / car_stalls_module
716         #print('Vertical # of modules: ',modules_hc)
717         #falta la z y saber si es factible, eso se hace con el
               module width (la tabla)
718
719       #print('Cars which are to be parked')
720   #-----------------------------------------------------
721   #Number of car stalls per module row (parking spots)
722         y_hc = float(sd_hc * math.cos(radianes_pa_hc))
```

```
723         x_hc = math.floor((md - y_hc) / pw_hc)
724         #md_r_hc = pw_hc * x_hc + y_hc
725         #cs_per_module = x_hc * mw_hc
726         #module_hc_2 = demand_hc / cs_per_module
727
728         module_hc_2 = float(0)
729         md_r_hc = float(0)
730
731         cs_per_module = float(0)
732
733         mwtabla_hc = float(0)
734
735         print('Horizontal # of modules: ',module_hc_2)
736         print('Module Depth required = ',md_r_hc)
737         #cs_per_module = x_hc * mw_hc
738         print('Handicapped car stalls per module = ',cs_per_module
                )
739         print('Module Width: ',mwtabla_hc)
740
741     elif hc == 'no':
742
743         module_hc_2 = float(0)
744         md_r_hc = float(0)
745         print('Module Depth required = ',md_r_hc)
746         cs_per_module = float(0)
747         print('Handicapped car stalls per module = ',cs_per_module
                )
748         mwtabla_hc = float(0)
```

```
749         print('Module Width: ',mwtabla_hc)
750         mw_hc = float(0)
751         pw_hc = 0
752
753     #-----------------------------------------------------
754     #comenzamos con los autos compactos
755
756        c = input("Compact? (yes/no) ")
757        if c == 'yes':
758           hm_c = input('How much? ')
759           sw_c = input('Choose the stall width: ') #preguntar si el
                  sw siempre es de 8'' para Compact cars
760           w_c = input('Module type (W): ')
761           #rows_c = input('Rows: ')
762           pa_c = input('Parking angle: ')
763           sd_c = input('Stall Depth (SD): ')
764           mwtabla_c1 = input('Check the table and add the feet
                  corresponding to parameter given: ')
765           mwtabla_c2 = input("Check the table and add the inches
                  corresponding to parameter given: ")
766           mwtabla_c1 = float(mwtabla_c1)
767           mwtabla_c2 = float(mwtabla_c2)
768           mwtabla_c = float(mwtabla_c1 + (mwtabla_c2/12))
769
770     #-----------------------------------------------------
771     #convertimos los inputs en numeros decimales
772
773           hm_c = float(hm_c)
```

```
774         sw_c = float(sw_c)
775         #rows_c = float(rows_c)
776         pa_c = float(pa_c)
777         sd_c = float(sd_c)
778
779
780     #numero de filas por W
781         if w_c == 'W1':
782             mw_c = float(1)
783
784         elif w_c == 'W2':
785             mw_c = float(2)
786
787         elif w_c == 'W3':
788             mw_c = float(2)
789
790         elif w_c == 'W4':
791             mw_c = float(2)
792
793     #-------------------------------------------------
794
795         print('Table. Parking layout alternative
                ----------------------------------------------')
796         import math
797         #print('Parking angle: ', pa_c)
798         demand_c = float(demand * hm_c)
799         redondear_demand_c = float(math.ceil(demand_c))
```

```
800         print('Demand (units): ',redondear_demand_c) #preguntar si
                se redondea hacia arriba o solo redondear
801         print('Parking angle (): ', pa_c)
802         print('Module type (W): ',w_c)
803         print('SW (feet): ',sw_c)
804         #import numpy
805         #radiandes_pa_hc = ((pa_hc * numpy.pi) / 180)
806         radianes_pa_c = float(math.radians(pa_c))
807         pw_c = float(sw_c / math.sin(radianes_pa_c))
808         print('PW (feet): ',pw_c)
809         #para handicapped no se pone numero de modulos ni module
               width
810         #stall_d_c = float(sd_c * math.cos(radianes_pa_c))
811         #x_per_row_c = (md / pw_c) + stall_d_c
812         #entero_x_per_row_c = int(x_per_row_c)
813         #if x_per_row_c >= entero_x_per_row_c:
814         # x_per_row_c = entero_x_per_row_c
815         #car_stalls_module_c = x_per_row_c * rows_c
816         #modules_c = demand_c / car_stalls_module_c
817         #print('Vertical # of modules: ',modules_c)
818         #falta la z y saber si es factible
819     #-------------------------------------------------------
820     #Number of car stalls per module row (parking spots)
821         y_c = float(sd_c * math.cos(radianes_pa_c))
822         x_c = float(math.floor((md - y_c) / pw_c))
823         md_r_c = float(pw_c * x_c + y_c)
824         cs_per_module_c = float(x_c * mw_c)
825         module_c_2 = (math.ceil(demand_c / cs_per_module_c))
```

```
826         module_c_2 = float(module_c_2)
827         print('y= ', y_c)
828         print('x= ', x_c)
829         print('Horizontal # modules: ',module_c_2)
830         print('Module Depth required = ',md_r_c)
831         print('Compact car stalls per module = ',cs_per_module_c)
832         print('Module Width (feet): ',mwtabla_c)
833
834      elif c == 'no':
835
836         module_c_2 = float(0)
837         md_r_c = float(0)
838         print('Module Depth required = ',md_r_c)
839         cs_per_module_c = float(0)
840         print('Handicapped car stalls per module = ',
               cs_per_module_c)
841         mwtabla_c = float(0)
842         print('Module Width (feet): ',mwtabla_c)
843         mw_c = float(0)
844
845
846
847   #-----------------------------------------------
848
849   #autos standard
850      s = input("Standard? (yes/no) ")
851
852      if s == 'yes':
```

```
853         hm_s = input('How much? ( %) ')
854         sw_s = input('Choose the stall width (SW (feet)): ')
855         w_s = input('Module type (W): ')
856         #rows_s = input('Rows: ')
857         pa_s = input('Parking angle ():')
858         sd_s = input('Stall Depth (SD (feet)): ')
859         mwtabla_s1 = input('Check the table and add the feet
               corresponding to parameter given: ')
860         mwtabla_s2 = input("Check the table and add the inches
               corresponding to parameter given: ")
861         mwtabla_s1 = float(mwtabla_s1)
862         mwtabla_s2 = float(mwtabla_s2)
863         mwtabla_s = float(mwtabla_s1 + (mwtabla_s2/12))
864
865   #-------------------------------------------------
866   #convertimos los inputs en numeros decimales
867         hm_s = float(hm_s)
868         sw_s = float(sw_s)
869         #rows_s = float(rows_s)
870         pa_s = float(pa_s)
871         sd_s = float(sd_s)
872
873   #numero de filas por W
874         if w_s == 'W1':
875            mw_s = float(1)
876
877         elif w_s == 'W2':
878            mw_s = float(2)
```

```
879
880         elif w_s == 'W3':
881             mw_s = float(2)
882
883         elif w_s == 'W4':
884             mw_s = float(2)
885
886     #-----------------------------------------------------
887         print('Table. Parking layout alternative
                ------------------------------------------')
888         import math
889         #print('Parking angle (): ', pa_s)
890         demand_s = float(demand * hm_s)
891         redondear_demand_s = float(math.ceil(demand_s))
892         print('Demand (units): ',redondear_demand_s) #preguntar si
                se redondea hacia arriba o solo redondear
893         print('Parking angle (): ', pa_s)
894         print('Module type (W): ',w_s)
895         print('SW (feet): ',sw_s)
896         #import numpy
897         #radiandes_pa_hc = ((pa_hc * numpy.pi) / 180)
898         radianes_pa_s = float(math.radians(pa_s))
899         pw_s = float(sw_s / math.sin(radianes_pa_s))
900         print('PW (feet): ',pw_s)
901         #para handicapped no se pone numero de modulos ni module
                width
902         #stall_d_s = float(sd_s * math.cos(radianes_pa_s))
903         #x_per_row_s = (md / pw_s) + stall_d_s
```

```
904         #entero_x_per_row_s = int(x_per_row_s)
905         #if x_per_row_s >= entero_x_per_row_s:
906         # x_per_row_s = entero_x_per_row_s
907         #car_stalls_module_s = x_per_row_s * rows_s
908         #modules_s = demand_s / car_stalls_module_s
909         #print('Vertical # of modules: ',modules_s)
910         #falta la z y saber si es factible
911   #-----------------------------------------------------
912   #Number of car stalls per module row (parking spots)
913         y_s = float(sd_s * math.cos(radianes_pa_s))
914         x_s = float(math.floor((md - y_s) / pw_s))
915         md_r_s = float(pw_s * x_s + y_s)
916         cs_per_module_s = float(x_s * mw_s)
917         module_s_2 = (math.ceil(demand_s / cs_per_module_s))
918         module_s_2 = float(module_s_2)
919         print('Horizontal # of modules: ',module_s_2)
920         print('Module Depth required = ',md_r_s)
921         print('Handicapped car stalls per module = ',
               cs_per_module_s)
922         print('Module Width (feet): ',mwtabla_s)
923
924      elif s == 'no':
925
926         module_s_2 = float(0)
927         md_r_s = float(0)
928         print('Module Depth required = ',md_r_s)
929         cs_per_module_s = float(0)
```

```
930         print('Handicapped car stalls per module = ',
               cs_per_module_s)
931         mwtabla_s = float(0)
932         print('Module Width (feet): ',mwtabla_s)
933         mw_s = float(0)
934
935
936     #-----------------------------------------------------
937     #Large
938
939        l = input("Large? (yes/no) ")
940
941        if l == 'yes':
942           hm_l = input('How much? ( %) ')
943           sw_l = input('Choose the stall width (SW (feet)): ')
944           w_l = input('Module type (W): ')
945           #rows_l = input('Rows: ')
946           pa_l = input('Parking angle ():')
947           sd_l = input('Stall Depth (SD (feet)): ')
948           mwtabla_l1 = input('Check the table and add the feets
                 corresponding to parameter given: ')
949           mwtabla_l2 = input("Check the table and add the inches
                 corresponding to parameter given: ")
950           mwtabla_l1 = float(mwtabla_l1)
951           mwtabla_l2 = float(mwtabla_l2)
952           mwtabla_l = float(mwtabla_l1 + (mwtabla_l2/12))
953
954     #-----------------------------------------------------
```

```
955    #convertimos los inputs en numeros decimales
956          hm_l = float(hm_l)
957          sw_l = float(sw_l)
958          #rows_l = float(rows_l)
959          pa_l = float(pa_l)
960          sd_l = float(sd_l)
961
962    #numero de filas por W
963          if w_l == 'W1':
964             mw_l = float(1)
965
966          elif w_l == 'W2':
967             mw_l = float(2)
968
969          elif w_l == 'W3':
970             mw_l = float(2)
971
972          elif w_l == 'W4':
973             mw_l = float(2)
974
975    #-------------------------------------------------
976          print('Table. Parking layout alternative
                ----------------------------------------------')
977          import math
978          print('Parking angle (): ', pa_l)
979          demand_l = float(demand * hm_l)
980          redondear_demand_l = math.ceil(demand_l)
```

```
981         print('Demand: ',redondear_demand_l) #preguntar si se
               redondea hacia arriba o solo redondear
982         print('Parking angle (): ', pa_l)
983         print('Module type (W): ',w_l)
984         print('SW (feet): ',sw_l)
985         #import numpy
986         #radiandes_pa_hc = ((pa_hc * numpy.pi) / 180)
987         radianes_pa_l = float(math.radians(pa_l))
988         pw_l = float(sw_l / math.sin(radianes_pa_l))
989         print('PW (feet): ',pw_l)
990         #para handicapped no se pone numero de modulos ni module
               width
991         #stall_d_l = float(sd_l * math.cos(radianes_pa_l))
992         #x_per_row_l = (md / pw_l) + stall_d_l
993         #entero_x_per_row_l = int(x_per_row_l)
994         #if x_per_row_l >= entero_x_per_row_l:
995         # x_per_row_l = entero_x_per_row_l
996         #car_stalls_module_l = x_per_row_l * rows_l
997         #modules_l = demand_l / car_stalls_module_l
998         #print('Vertical # of modules: ',modules_l)
999         #falta la z y saber si es factible
1000
1001    #-------------------------------------------------------
1002
1003    #Number of car stalls per module row (parking spots)
1004          y_l = float(sd_l * math.cos(radianes_pa_l))
1005          x_l = float(math.floor((md - y_l) / pw_l))
1006          md_r_l = float(pw_l * x_l + y_l)
```

```
1007            cs_per_module_l = float(x_l * mw_l)
1008            module_l_2 = (math.ceil(demand_l / cs_per_module_l))
1009            module_l_2 = float(module_l_2)
1010            print('Horizontal # of module: ',module_l_2)
1011            print('Module Depth required (feet) = ',md_r_l)
1012            print('Handicapped car stalls per module = ',
                   cs_per_module_l)
1013            print('Module Width (feet): ',mwtabla_l)
1014
1015        elif l == 'no':
1016
1017            module_l_2 = float(0)
1018            md_r_l = float(0)
1019            print('Module Depth required (feet) = ',md_r_l)
1020            cs_per_module_l = float(0)
1021            print('Large car stalls per module = ',cs_per_module_l)
1022            mwtabla_l = float(0)
1023            print('Module Width (feet): ',mwtabla_l)
1024            mw_l = float(0)
1025
1026
1027    #-----------------------------------------------------
1028    #Feasible?
1029        z = float((module_hc_2 * mwtabla_hc) + (module_c_2 *
               mwtabla_c) + (module_s_2 * mwtabla_s) + (module_l_2 *
               mwtabla_l))
1030        print('Z = ',z)
1031        print(space_vertical)
```

```
1032        if space_vertical >= z:
1033            print('Factible')
1034        else:
1035            print('No factible')
1036    #-----------------------------------------------------
1037    #imagen y sketch parte 1
1038
1039        if space_vertical >= z:
1040
1041            print("Draw an sketch similar to the image depending on # of modules of each type of car")
1042
1043            from PIL import Image
1044
1045            # creating a object
1046            im_vertical = Image.open('vertical.png')
1047
1048            im_vertical.show()
1049
1050            print('Consider external communication channels as in the image ')
1051            im_vertical_channels = Image.open('channels.png')
1052            im_vertical_channels.show()
1053
1054            print('Consider entrance and exit ')
1055            im_vertical_ee = Image.open('entrance and exit.png')
1056            im_vertical_ee.show()
1057
```

```
1058          print('Consider␣circulation␣lines␣in␣the␣sketch␣')
1059          im_vertical_circulation = Image.open('circulation␣lines.
                 png')
1060          im_vertical_circulation.show()
1061
1062          print('Consider␣handicapped␣spaces␣according␣to␣the␣table␣
                 and␣the␣area␣you␣want␣to␣add␣them')
1063          im_vertical_hcspaces = Image.open('handicapped␣spaces.png'
                 )
1064          im_vertical_hcspaces.show()
1065
1066       #calculos de las filas
1067          total_rows = mw_hc * module_hc_2 + mw_c * module_c_2 +
                 mw_s * module_s_2 + mw_l * module_l_2 #el de
                 handicapped no es necesario, solo lo puse porque en
                 este baso todo pero los dems hay que agregarlos y
                 sumarlos
1068          print('Total␣rows:␣',total_rows)
1069          circulation_lines = input('How␣many␣feet␣for␣circulation␣
                 lines?␣')
1070          circulation_lines = float(circulation_lines)
1071
1072          #tomar en cuenta poner un if para si autos compactos/
                 standard/large vertical # of modules >= 1
1073
1074
1075       #para hc
1076       # if module_hc_2 >= 1:
```

```
1077        # print('For rows in Handicapped cars ')
1078        # count_hc = 0
1079              #lista_hc = []
1080              #lista_hc = append(count_hc)
1081              #for num in lista_hc:
1082        # while count_hc <= module_hc_2:
1083        # spacehc_hc = input('Are there any spaces for handicapped
                cars? ')
1084        # if spacehc_hc == 'yes':
1085        # spacehc_h_hc = input('How many? ')
1086        # spacehc_h_hc = float(spacehc_h_hc)
1087        # elif spacehc_hc == 'no':
1088        # spacehc_h_hc = 0
1089        # spacehc_h_hc = float(spacehc_h_hc)
1090        # spacecl_hc = input('Are there any spaces for cirulation
                lines? ')
1091        # if spacecl_hc == 'yes':
1092        # spacecl_h_hc = input('How many? ')
1093        # spacecl_h_hc = float(spacecl_h_hc)
1094        # #spacecl_hi_hc = input('How many inches of each circulation
                 line? ')
1095        # #spacecl_hi_hc = float(spacecl_hi_hc)
1096        # elif spacecl_hc == 'no':
1097        # spacecl_h_hc = 0
1098        # spacecl_h_hc = float(spacecl_h_hc)
1099        #
1100        #
```

```
1101        # row_hc = float((md - (spacehc_h_hc * pw_hc) - (spacecl_h_hc
                * circulation_lines) - y_hc) / pw_hc)
1102        # print('Row ',count_hc, '=', row_hc)
1103
1104        # count_hc = count_hc + 1
1105
1106        #para compact rows
1107
1108           if module_c_2 >= 1:
1109              #countmodule_c = 1
1110              lista_c = []
1111              lista_hc_c = []
1112              lista_row_c = []
1113              count_c = 1
1114
1115              count_r_c = 1
1116              #lista_hc_c = append(spacehc_h_c)
1117              print('For␣rows␣in␣compact␣cars␣')
1118              while count_r_c <= (mw_c * module_c_2):
1119                 spacehc_c = input('Are␣there␣any␣spaces␣for␣
                        handicapped␣cars?␣')
1120                 if spacehc_c == 'yes':
1121                    spacehc_h_c = input('How␣many?␣')
1122                    spacehc_h_c = float(spacehc_h_c)
1123                 elif spacehc_c == 'no':
1124                    spacehc_h_c = 0
1125                    spacehc_h_c = float(spacehc_h_c)
```

```
1126                spacecl_c = input('Are there any spaces for
                       cirulation lines? ')
1127                if spacecl_c == 'yes':
1128                   spacecl_h_c = input('How many? ')
1129                   spacecl_h_c = float(spacecl_h_c)
1130                         #spacecl_hi_hc = input('How many inches of
                              each circulation line? ')
1131                         #spacecl_hi_hc = float(spacecl_hi_hc)
1132                elif spacecl_c == 'no':
1133                  spacecl_h_c = 0
1134                  spacecl_h_c = float(spacecl_h_c)
1135
1136
1137                row_c = float((md - (spacehc_h_c * pw_hc) - (
                       spacecl_h_c * circulation_lines) - y_c) / pw_c)
1138                print('Row ',count_r_c, '=', row_c)
1139                lista_row_c.append(row_c)
1140                lista_c.append(count_r_c)
1141                lista_hc_c.append(spacehc_h_c)
1142
1143                count_r_c = count_r_c + 1
1144             count_c = count_c + 1
1145
1146
1147       #para standard
1148
1149          if module_s_2 >= 1:
1150             lista_s = []
```

```
1151            lista_hc_s = []
1152            lista_row_s = []
1153
1154            count_s = 1
1155
1156            print('For␣rows␣in␣standard␣cars␣')
1157               #count_s = 1
1158               #lista_s = []
1159               #lista_s.append(count_s)
1160
1161               #lista_hc_s = []
1162               #lista_row_1 = []
1163               #lista_hc = []
1164               #lista_hc = append(count_hc)
1165               #for num in lista_hc:
1166            count_r_s = 1
1167            #lista_hc_s = append(spacehc_h_s)
1168            while count_r_s <= (mw_s * module_s_2):
1169               #lista_hc_s = append(spacehc_h_s)
1170               spacehc_s = input('Are␣there␣any␣spaces␣for␣
                     handicapped␣cars?␣')
1171               if spacehc_s == 'yes':
1172                  spacehc_h_s = input('How␣many?␣')
1173                  spacehc_h_s = float(spacehc_h_s)
1174               elif spacehc_s == 'no':
1175                  spacehc_h_s = 0
1176                  spacehc_h_s = float(spacehc_h_s)
```

```
1177                spacecl_s = input('Are there any spaces for
                       cirulation lines? ')
1178                if spacecl_s == 'yes':
1179                   spacecl_h_s = input('How many? ')
1180                   spacecl_h_s = float(spacecl_h_s)
1181                      #spacecl_hi_hc = input('How many inches of
                             each circulation line? ')
1182                      #spacecl_hi_hc = float(spacecl_hi_hc)
1183                elif spacecl_s == 'no':
1184                   spacecl_h_s = 0
1185                   spacecl_h_s = float(spacecl_h_s)
1186
1187
1188                row_s = float((md - (spacehc_h_s * pw_hc) - (
                       spacecl_h_s * circulation_lines) - y_s) / pw_s)
1189                print('Row ',count_r_s, '=', row_s)
1190                lista_row_s.append(row_s)
1191                lista_s.append(count_s)
1192                lista_hc_s.append(spacehc_h_s)
1193                count_r_s = count_r_s + 1
1194             count_s = count_s + 1
1195
1196          #para large rows
1197          countmodule_l = 1
1198          if module_l_2 >= 1:
1199             lista_l = []
1200             lista_hc_l = []
1201             lista_row_l = []
```

```
1202            count_l = 1
1203
1204            print('For␣rows␣in␣large␣cars␣')
1205
1206               #lista_l = []
1207               #lista_l.append(count_l)
1208
1209               #lista_hc_l = []
1210               #lista_row_l = []
1211               #lista_hc = []
1212            #lista_hc = append(count_hc)
1213            count_r_l = 1
1214               #for num in lista_hc:
1215            while count_r_l <= (mw_l * module_l_2):
1216                  #lista_hc_l = append(spacehc_h_l)
1217               spacehc_l = input('Are␣there␣any␣spaces␣for␣
                     handicapped␣cars?␣')
1218               if spacehc_l == 'yes':
1219                  spacehc_h_l = input('How␣many?␣')
1220                  spacehc_h_l = float(spacehc_h_l)
1221               elif spacehc_l == 'no':
1222                  spacehc_h_l = 0
1223                  spacehc_h_l = float(spacehc_h_l)
1224               spacecl_l = input('Are␣there␣any␣spaces␣for␣
                     cirulation␣lines?␣')
1225               if spacecl_l == 'yes':
1226                  spacecl_h_l = input('How␣many?␣')
1227                  spacecl_h_l = float(spacecl_h_l)
```

```
1228                    #spacecl_hi_hc = input('How many inches of
                           each circulation line? ')
1229                    #spacecl_hi_hc = float(spacecl_hi_hc)
1230                elif spacecl_l == 'no':
1231                   spacecl_h_l = 0
1232                   spacecl_h_l = float(spacecl_h_l)
1233
1234
1235                row_l = float((md - (spacehc_h_l * pw_hc) - (
                       spacecl_h_l * circulation_lines) - y_l) / pw_l)
1236                print('Row ',count_r_l, '=', row_l)
1237                lista_row_l.append(row_l)
1238                lista_l.append(count_l)
1239                lista_hc_l.append(spacehc_h_l)
1240
1241                count_r_l = count_r_l + 1
1242             count_l = count_l + 1
1243
1244          print("
               Resumen_____________________________________________________
               ")
1245
1246          if hc == 'yes':
1247             print('Demand per handicapped cars: ',demand_hc)
1248             #for row_hc_c2 in lista_c:
1249             if module_c_2 >= 1:
1250                print('Compact car Row ',lista_hc_c,': ',lista_hc_c)
1251             #for row_hc_s2 in lista_s:
```

```
1252            if module_s_2 >= 1:
1253               print('Standard car Row ',lista_hc_s,': ',lista_hc_s
                      )
1254               #for row_hc_l2 in lista_l:
1255               if module_l_2 >= 1:
1256                  print('Standard car Row ',lista_hc_l,': ',lista_hc_l
                         )
1257
1258         if module_c_2 >= 1:
1259            print('Demand per compact cars: ',demand_c)
1260            #for row_c_c2 in lista_c:
1261            print (lista_c)
1262            print('Compact cars per Row ')
1263            print('Row: ',lista_c)
1264            print('Stalls: ',lista_row_c)
1265
1266         if module_s_2 >= 1:
1267            print('Demand per standard cars: ',demand_s)
1268            #for row_s_s2 in lista_s:
1269            print('Standard cars per Row ')
1270            print('Row: ',lista_s)
1271            print('Stalls: ',lista_row_s)
1272
1273         if module_l_2 >= 1:
1274            print('Demand per large cars: ',demand_l)
1275            #for row_c_c2 in lista_l:
1276            print('Large car per Row ')
1277            print('Row: ',lista_l)
```

```
1278            print('Stalls: ',lista_row_l)
1279
1280    count_s = count_s + 1
```

Referencias

Asmael, N. M., y Turky, G. F. (2022). Parking requirement of institutional land use. En *Iop conference series: Earth and environmental science* (Vol. 961, cap. 1). Descargado de www.scopus.com

Board, H. R. (1971). *Principles, parking.* Washington, DC. Descargado de https://onlinepubs .trb.org/Onlinepubs/sr/sr125.pdf

Leonard, B. G., May, J. F., y Pringle Jr, P. (s.f.). Ia summary repoti employment center parking facilities.

Litman, T. (2016). *Parking management: strategies, evaluation and planning*. Victoria Transport Policy Institute Victoria, BC, Canada.

McConochie, W. R. (1961). Design of parking lots and garages.

Moreno Rincón, M., y cols. (2017). *Análisis de los beneficios económicos y sociales percibidos con la puesta en marcha de cacique el centro comercial y de negocios ph en bucaramanga y su área metropolitana* (Tesis Doctoral no publicada). Universidad EAFIT.

Nikolaev, D. I., Chubur, N. V., Krasnoschekov, V. V., y Diuldin, M. V. (2021). Smart parking for an ecological type of transport-electric scooter. En *Journal of physics: Conference series* (Vol. 1942, cap. 1). Descargado de www.scopus.com (Cited By :1)

Normas técnicas complementarias del regramento de ordenamiento territorial (rot) del municipio de celaya, gto. para el proyecto arquitectónico. (2006). Descargado de http://www.celaya.gob.mx/cya/wp-content/uploads/2017/08/norma_tecnica_proyecto _arquitectonicoNorma-Tecnica-del-Proyecto-Arquitectonico.pdf

Ramsey, C. G., y Sleeper, H. R. (2011). *Architectural graphic standards* (Vol. 8). John Wiley & Sons.

Secretaría de gobernación. (2006). *Proyecto de norma mexicana proy-nmx-r-050-scfi-2005, accesibilidad de las personas con discapacidad a espacios construidos de servicio al público-específicaciones de seguridad.* Descargado de http://dof.gob.mx/nota_detalle.php ?codigo=4913218&fecha=13/03/2006

Secretaría de gobernación. (2020). *Proyecto de norma oficial mexicana proy-nom-001-sedatu-2020.* Descargado de https://www.dof.gob.mx/nota_detalle.php?codigo= 5608336&fecha=21/12/2020

Tompkins, J. A., White, J. A., y Bozer, J. M. A., Javuz A. Tanchoco. (2010). *Facilities planning fourth edition.* John Wiley Sons.