Queue

Queue visually represents the increasing (and fascinating) intertwinement of technology and humanity in typeface form. Stylistically it incorporates characteristics from type styles engineered to be read by machines, shapes commonly seen in fonts designed to make it easier for humans to read computer code and the proportions and rhythm of humanist calligraphy. The family has open, legible, precise letterforms that look great on screen as well as in print, especially in small sizes.

Thin
Thin Italic
Light
Light Italic
Book
Book Italic
Medium
Medium Italic
Bold
Bold Italic
Black
Black Italic

DESIGNER
TAL LEMING
2014

12 STYLES
6 WEIGHTS
ROMAN AND ITALIC

Type Supply

DIMORPHIC

Nutrition Principles and Clinical Nutrition

CONSCIOUS

Foundations for Programming Languages

EXHIBITED

Human-Computer Interaction Handbook

OUTWEIGH

Eleven Modern Pioneers of Archaeology

NEOPRENE

Genome Sequencing for the Rest of Us

SCORPION

Beginnings of Machine Cryptography

GEOMETRIC

Plant Viral Vectors for Protein Expression

BIOSENSOR

Computer Architecture and Organization

CINEMATIC

The Complete Guide to Sports Nutrition

MONOXIDE

An Introduction to Genetic Algorithms

GREMLINS

Corporatization of Agricultural Policy

DIATOMIC

Electron Sharing and Covalent Bonds

INBOUND
CHLORINE
MECHANICS
ORDINOGRAM
NANOMEDICINE
SUBMICROMETER
DIHYDROCHLORIDE
CHEMOINFORMATICS
INTERDISCIPLINARITY

SUNBURN
MODIFIED
ORGANISMS
PREDEFINING
COMPUTERIZED
NEUROINTENSIVE
GEOVISUALIZATION
INTERRELATIONSHIP
CHEMOORGANOTROPHS

Hominids
Generated
Mainstream
Somatostatin
Postmodernism
Cryptosporidiosis
Japanese-American
Systems Engineering
History & Mathematics

Chemists
Nematode
Glutathione
Photochromic
Object Oriented
Neuroinformatics
Self-Aware Systems
Electrocorticography
Continuous Integration

In computer science, a queue is a collection of entities that are maintained in a *sequence* and can be modified by the addition of entities at one end of the sequence and the removal of entities from the other end of the sequence. By convention, the *end of the sequence at which elements are added is called the back, tail, or rear of the queue, and the end at which elements are removed is called the head*

The operation of adding an element to the rear of the queue is known as *enqueue*, and the operation of removing an element from the front is known as *dequeue*. Other operations may also be allowed, often including a peek or front operation that returns the value of the next element to be dequeued without dequeuing it. The operations *of a queue make it a first in first out (FIFO) data structure. In a FIFO data structure, the first element added to the queue will be the first one to be removed. This is equivalent to the requirement that once a new element is added, all elements*

OPSONIN
DISCRETE
GRADIENTS
NANOSECOND
SUBCONSCIOUS
MECHANICSBURG
COMPUTER-HUMAN
NEUROIMMUNOLOGY
SCIENTIFIC AMERICAN

HORIZON
CRINOIDS
MANIFESTO
SEMIOTICIAN
BIOIMPEDANCE
COMMUNICATING
PEPTIDOMIMETICS
OLIGODENDROCYTIC
RETROTRANSPOSITION

Selenium
Education
Cleanrooms
Instantiation
Camerini-Otero
Nanoremediation
Genome Dictionary
Human Development
Quantum Computation

Baconian
Converted
Microdrives
Senthilkumar
Documentation
Glottochronology
Macromanagement
CTO Network Library
A Personal Perspective

In computer science, a queue is a collection of entities that are maintained in a *sequence* and can be modified by the addition of entities at one end of the sequence and the removal of entities from the other end of the sequence. By convention, the *end of the sequence at which elements are added is called the back, tail, or rear of the queue, and the end at which elements are removed is called the*

The operation of adding an element to the rear of the queue is known as *enqueue,* and the operation of removing an element from the front is known as *dequeue.* Other operations may also be allowed, often including a peek or front operation that returns the value of the next element to be dequeued without dequeuing it. The operations *of a queue make it a first in first out (FIFO) data structure. In a FIFO data structure, the first element added to the queue will be the first one to be removed. This is equivalent to the requirement that once a new element is added,* IN COMPUTER SCIENCE, A QUEUE IS A COLLECTION OF ENTITIES that are maintained in a *sequence* and can be modified by the addition of entities at one end of the sequence and the removal of entities from the other end of the sequence. By convention, the end of the sequence at which elements are added is called the back, tail, or rear of the queue, and the end at which elements are removed is called the head or front of the queue,

IN A *PRIORITY QUEUE,* AN ELEMENT WITH HIGH PRIORITY IS SERVED BEFORE AN ELEMENT WITH LOW PRIORITY. IN SOME IMPLEMENTATIONS, IF TWO ELEMENTS HAVE THE SAME PRIORITY, THEY ARE SERVED ACCORDING TO THE ORDER IN WHICH THEY WERE ENQUEUED, WHILE IN OTHER IMPLEMENTATIONS, ORDERING OF ELEMENTS WITH THE SAME PRIORITY IS UNDEFINED. WHILE PRIORITY QUEUES *ARE OFTEN IMPLEMENTED WITH HEAPS, THEY ARE CONCEPTUALLY DISTINCT FROM HEAPS. A PRIORITY QUEUE IS A CONCEPT LIKE "A LIST" OR "A MAP"; JUST AS A LIST CAN BE IMPLEMENTED WITH A LINKED LIST OR AN ARRAY, A PRIORITY QUEUE CAN BE*

The operation of adding an element to the rear of the queue is known as enqueue, *and the operation of removing an element from the front is known as* dequeue. *Other operations may also be allowed, often including a peek or front operation that returns the value of the next element to be dequeued without dequeuing it. The operations of a queue make it a first in first out (FIFO) data structure. In a FIFO data structure, the*

In a FIFO data structure, the first element added to the queue will be the first one to be removed. This is equivalent to the requirement that once a new element is added, all elements that were added before have to be removed before the new element can be removed. A queue is an example of a linear data structure, or more abstractly a sequential collection. *Queues are common in computer programs, where they are implemented as data structures coupled with access routines, as an abstract data structure or in object-oriented languages as classes*. Common implementations are circular buffers and linked lists. Queues provide services in computer science, transport, and operations research where various entities such as data, objects, persons, or events are stored and held to be processed later. In these contexts, the queue performs the function of a buffer. Another usage of queues is in the implementation of breadth-first search. Theoretically, one characteristic of a queue is that it does not have a specific capacity. Regardless of how many elements are already contained, a new element can always be added. It can also be empty, at which point removing an element will be impossible until a new element has been added again. Fixed-length arrays are limited in capacity, but it is not true that items need to be copied towards the head of the queue. The simple trick of turning the array into a closed circle and letting the head and tail drift around endlessly in

The simple trick of turning the array into a closed circle and letting the head and tail drift around endlessly in that circle makes it unnecessary to ever move items stored in the array. If *n* is the size of the array, then computing indices modulo *n* will turn the array into a circle. This is still the conceptually simplest way to construct a queue in a high-level language, but it does admittedly slow things down a little, because the array indices must be compared to zero and the array size, which is comparable to the time taken to check whether an array index is out of bounds, which some languages do, but this will certainly be the method of choice for a quick and dirty implementation, or for any high-level language that does not have pointer syntax. The array size must be declared ahead of time, but some implementations simply double the declared array size when overflow occurs. Most modern languages with objects or pointers can implement or come with libraries for dynamic lists. Such data structures may have not specified a fixed capacity limit besides memory constraints. Queue overflow results from trying to add an element onto a full queue and queue underflow happens when trying to remove an element from an empty queue. A *bounded queue* is a queue limited to a fixed number of items. There are several efficient implementations of FIFO queues. An efficient implementation is one that can perform the operations—enqueuing and dequeuing—in O(1) time. In computer science, a priority queue is an abstract data type similar to regular queue

METRICS
SUMMING
DISPERSED
GREENSTONE
NEOHUMANISM
CONTAMINATION
MACROSTRUCTURE
SUPER-RESOLUTION
BEOWULF DEFINITION

ORATION
IONIZING
CHIPTUNES
HAUSTORIUM
STRIGIFORMES
DECOMPOSITION
COUNTERMEASURE
INTROSPECTIONISM
SCIENCE-TECHNOLOGY

Microbes
Orchestra
Ecotourism
Cuculiformes
Nucleoskeleton
One-Dimensional
Multiprogramming
Convergence Review
Ecommerce News Blog

Gerardus
Miniature
Conundrum
Institutional
Stress-Induced
Foundationalism
Ophthalmologicals
Transcomputational
Compiler Construction

In computer science, a queue is a collection of entities that are maintained in a *sequence* and can be modified by the addition of entities at one end of the sequence and the removal of entities from the other end of the sequence. By convention, *the end of the sequence at which elements are added is called the back, tail, or rear of the queue, and the end at which elements are removed is*

The operation of adding an element to the rear of the queue is known as *enqueue,* and the operation of removing an element from the front is known as *dequeue*. Other operations may also be allowed, often including a peek or front operation that returns the value of the next element to be dequeued without dequeuing *it. The operations of a queue make it a first in first out (FIFO) data structure. In a FIFO data structure, the first element added to the queue will be the first one to be removed. This is equivalent to the requirement that once a*

IN COMPUTER SCIENCE, A QUEUE IS A COLLECTION OF entities that are maintained in a *sequence* and can be modified by the addition of entities at one end of the sequence and the removal of entities from the other end of the sequence. By convention, the end of the sequence at which elements are added is called the back, tail, or rear of the queue, and the end at which elements are removed is called the head or front of

IN A *PRIORITY QUEUE,* AN ELEMENT WITH HIGH PRIORITY IS SERVED BEFORE AN ELEMENT WITH LOW PRIORITY. IN SOME IMPLEMENTATIONS, IF TWO ELEMENTS HAVE THE SAME PRIORITY, THEY ARE SERVED ACCORDING TO THE ORDER IN WHICH THEY WERE ENQUEUED, WHILE IN OTHER IMPLEMENTATIONS, ORDERING OF ELEMENTS WITH THE SAME PRIORITY IS UNDEFINED. WHILE PRIORITY QUEUES *ARE OFTEN IMPLEMENTED WITH HEAPS, THEY ARE CONCEPTUALLY DISTINCT FROM HEAPS. A PRIORITY QUEUE IS A CONCEPT LIKE "A LIST" OR "A MAP"; JUST AS A LIST CAN BE IMPLEMENTED WITH A LINKED LIST OR AN ARRAY, A PRIORITY QUEUE*

The operation of adding an element to the rear of the queue is known as enqueue, *and the operation of removing an element from the front is known as* dequeue. *Other operations may also be allowed, often including a peek or front operation that returns the value of the next element to be dequeued without dequeuing it. The operations of a queue make it a first in first out (FIFO) data structure. In a FIFO*

In a FIFO data structure, the first element added to the queue will be the first one to be removed. This is equivalent to the requirement that once a new element is added, all elements that were added before have to be removed before the new element can be removed. A queue is an example of a linear data structure, or more abstractly a sequential collection. *Queues are common in computer programs, where they are implemented as data structures coupled with access routines, as an abstract data structure or in object-oriented languages as classes.* Common implementations are circular buffers and linked lists. Queues provide services in computer science, transport, and operations research where various entities such as data, objects, persons, or events are stored and held to be processed later. In these contexts, the queue performs the function of a buffer. Another usage of queues is in the implementation of breadth-first search. Theoretically, one characteristic of a queue is that it does not have a specific capacity. Regardless of how many elements are already contained, a new element can always be added. It can also be empty, at which point removing an element will be impossible until a new element has been added again. Fixed-length arrays are limited in capacity, but it is not true that items need to be copied towards the head of the queue. The simple trick of turning the array into a closed circle and letting

The simple trick of turning the array into a closed circle and letting the head and tail drift around endlessly in that circle makes it unnecessary to ever move items stored in the array. If *n* is the size of the array, then computing indices modulo *n* will turn the array into a circle. This is still the conceptually simplest way to construct a queue in a high-level language, but it does admittedly slow things down a little, because the array indices must be compared to zero and the array size, which is comparable to the time taken to check whether an array index is out of bounds, which some languages do, but this will certainly be the method of choice for a quick and dirty implementation, or for any high-level language that does not have pointer syntax. The array size must be declared ahead of time, but some implementations simply double the declared array size when overflow occurs. Most modern languages with objects or pointers can implement or come with libraries for dynamic lists. Such data structures may have not specified a fixed capacity limit besides memory constraints. Queue overflow results from trying to add an element onto a full queue and queue underflow happens when trying to remove an element from an empty queue. A *bounded queue* is a queue limited to a fixed number of items. There are several efficient implementations of FIFO queues. An efficient implementation is one that can perform the operations—enqueuing and dequeuing—in O(1) time. In computer science, a priority queue is an abstract data type similar

GENETIC
BINOMEN
8,132,756.5
FUNCTIONED
OSTOMACHION
RECOMMENDING
COMPUTER–MUSIC
INSTRUMENTALISM
SECOND-GENERATION

BIOTECH
ORDERED
NUMEROUS
GENERATION
MEMORANDUM
SYMBIOGENESIS
BIOCOMPUTATION
GROUND-BREAKING
NEUROTHERAPEUTICS

Ornstein
Domestic
Circulation
Homeostasis
Generationism
Electrotechnical
Stereolithography
Hydro-Environment
Carbon Sequestration

Elevated
Samhitas
Memorable
Contribution
Nanomachines
9382 Circular St.
Hypermethylation
Stanford University
Bacterial Conjugation

In computer science, a queue is a collection of entities that are maintained in a *sequence* and can be modified by the addition of entities at one end of the sequence and the removal of entities from the other end of the sequence. By convention, *the end of the sequence at which elements are added is called the back, tail, or rear of the queue, and the end at which elements are removed*

The operation of adding an element to the rear of the queue is known as *enqueue,* and the operation of removing an element from the front is known as *dequeue*. Other operations may also be allowed, often including a peek or front operation that returns the value of the next element to be dequeued without *dequeuing it. The operations of a queue make it a first in first out (FIFO) data structure. In a FIFO data structure, the first element added to the queue will be the first one to be removed. This is equivalent to the requirement*

IN COMPUTER SCIENCE, A QUEUE IS A COLLECTION OF entities that are maintained in a *sequence* and can be modified by the addition of entities at one end of the sequence and the removal of entities from the other end of the sequence. By convention, the end of the sequence at which elements are added is called the back, tail, or rear of the queue, and the end at which elements are removed

IN A *PRIORITY QUEUE*, AN ELEMENT WITH HIGH PRIORITY IS SERVED BEFORE AN ELEMENT WITH LOW PRIORITY. IN SOME IMPLEMENTATIONS, IF TWO ELEMENTS HAVE THE SAME PRIORITY, THEY ARE SERVED ACCORDING TO THE ORDER IN WHICH THEY WERE ENQUEUED, WHILE IN OTHER IMPLEMENTATIONS, ORDERING OF ELEMENTS WITH THE SAME PRIORITY IS UNDEFINED. WHILE PRIORITY QUEUES *ARE OFTEN IMPLEMENTED WITH HEAPS, THEY ARE CONCEPTUALLY DISTINCT FROM HEAPS. A PRIORITY QUEUE IS A CONCEPT LIKE "A LIST" OR "A MAP"; JUST AS A LIST CAN BE IMPLEMENTED WITH A LINKED LIST OR AN ARRAY, A PRIORITY QUEUE*

The operation of adding an element to the rear of the queue is known as enqueue, *and the operation of removing an element from the front is known as* dequeue. *Other operations may also be allowed, often including a peek or front operation that returns the value of the next element to be dequeued without dequeuing it. The operations of a queue make it a first in first out (FIFO) data structure. In*

In a FIFO data structure, the first element added to the queue will be the first one to be removed. This is equivalent to the requirement that once a new element is added, all elements that were added before have to be removed before the new element can be removed. A queue is an example of a linear data structure, or more abstractly a sequential collection. *Queues are common in computer programs, where they are implemented as data structures coupled with access routines, as an abstract data structure or in object-oriented languages as classes.* Common implementations are circular buffers and linked lists. Queues provide services in computer science, transport, and operations research where various entities such as data, objects, persons, or events are stored and held to be processed later. In these contexts, the queue performs the function of a buffer. Another usage of queues is in the implementation of breadth-first search. Theoretically, one characteristic of a queue is that it does not have a specific capacity. Regardless of how many elements are already contained, a new element can always be added. It can also be empty, at which point removing an element will be impossible until a new element has been added again. Fixed-length arrays are limited in capacity, but it is not true that items need to be copied towards the head of the queue. The simple

The simple trick of turning the array into a closed circle and letting the head and tail drift around endlessly in that circle makes it unnecessary to ever move items stored in the array. If *n* is the size of the array, then computing indices modulo *n* will turn the array into a circle. This is still the conceptually simplest way to construct a queue in a high-level language, but it does admittedly slow things down a little, because the array indices must be compared to zero and the array size, which is comparable to the time taken to check whether an array index is out of bounds, which some languages do, but this will certainly be the method of choice for a quick and dirty implementation, or for any high-level language that does not have pointer syntax. The array size must be declared ahead of time, but some implementations simply double the declared array size when overflow occurs. Most modern languages with objects or pointers can implement or come with libraries for dynamic lists. Such data structures may have not specified a fixed capacity limit besides memory constraints. Queue overflow results from trying to add an element onto a full queue and queue underflow happens when trying to remove an element from an empty queue. A *bounded queue* is a queue limited to a fixed number of items. There are several efficient implementations of FIFO queues. An efficient implementation is one that can perform the operations—enqueuing and dequeuing—in O(1) time. In computer science, a

HUMANS
ORIGAMI
PHONEMIC
SUSTAINING
ECOCENTRISM
ORTHOGRAPHIC
MECHANOSENORS
CIRCUMSCRIPTION
PHOSPHOGLUCONATE

CELSIUS
DRAINER
OSHERSON
HYDRONIUM
GERSHENFELD
BIOCOMPUTING
CYBERSEMIOTICS
INTERCONVERSION
SUPERDETERMINISM

Isolated
Ouzounis
Duodenum
Chlorobiales
Normalization
Geombinatorics
Electronic Design
Operating Systems
The NASA Experience

Onboard
Hachette
Consisting
Plasmodium
Sadratnamala
Infrastructures
Cinematographer
Multiprogramming
Sequence-Dependent

In *computer science*, a queue is a collection of entities that are maintained in a *sequence* and can be modified by the addition of entities at one end of the sequence and the removal of entities from the other end of the sequence. By convention, *the end of the sequence at which elements are added is called the back, tail, or rear of the queue, and the end at which elements are*

The operation of adding an element to the rear of the queue is known as *enqueue*, and the operation of removing an element from the front is known as *dequeue*. Other operations may also be allowed, often including a peek or front operation that returns the value of the next element to be dequeued without *dequeuing it. The operations of a queue make it a first in first out (FIFO) data structure. In a FIFO data structure, the first element added to the queue will be the first one to be removed. This is equivalent to the requirement that*

IN COMPUTER SCIENCE, A QUEUE IS A COLLECTION OF entities that are maintained in a *sequence* and can be modified by the addition of entities at one end of the sequence and the removal of entities from the other end of the sequence. By convention, the end of the sequence at which elements are added is called the back, tail, or rear of the queue, and the end at which elements are removed

IN A *PRIORITY QUEUE*, AN ELEMENT WITH HIGH PRIORITY IS SERVED BEFORE AN ELEMENT WITH LOW PRIORITY. IN SOME IMPLEMENTATIONS, IF TWO ELEMENTS HAVE THE SAME PRIORITY, THEY ARE SERVED ACCORDING TO THE ORDER IN WHICH THEY WERE ENQUEUED, WHILE IN OTHER IMPLEMENTATIONS, ORDERING OF ELEMENTS WITH THE SAME PRIORITY IS UNDEFINED. WHILE PRIORITY QUEUES *ARE OFTEN IMPLEMENTED WITH HEAPS, THEY ARE CONCEPTUALLY DISTINCT FROM HEAPS. A PRIORITY QUEUE IS A CONCEPT LIKE "A LIST" OR "A MAP"; JUST AS A LIST CAN BE IMPLEMENTED WITH A LINKED LIST OR AN ARRAY, A PRIORITY QUEUE*

The operation of adding an element to the rear of the queue is known as enqueue, *and the operation of removing an element from the front is known as* dequeue. *Other operations may also be allowed, often including a peek or front operation that returns the value of the next element to be dequeued without dequeuing it. The operations of a queue make it a first in first out (FIFO)*

In a FIFO data structure, the first element added to the queue will be the first one to be removed. This is equivalent to the requirement that once a new element is added, all elements that were added before have to be removed before the new element can be removed. A queue is an example of a linear data structure, or more abstractly a sequential collection. *Queues are common in computer programs, where they are implemented as data structures coupled with access routines, as an abstract data structure or in object-oriented languages as classes.* Common implementations are circular buffers and linked lists. Queues provide services in computer science, transport, and operations research where various entities such as data, objects, persons, or events are stored and held to be processed later. In these contexts, the queue performs the function of a buffer. Another usage of queues is in the implementation of breadth-first search. Theoretically, one characteristic of a queue is that it does not have a specific capacity. Regardless of how many elements are already contained, a new element can always be added. It can also be empty, at which point removing an element will be impossible until a new element has been added again. Fixed-length arrays are limited in capacity, but it is not true that items need to be

The simple trick of turning the array into a closed circle and letting the head and tail drift around endlessly in that circle makes it unnecessary to ever move items stored in the array. If *n* is the size of the array, then computing indices modulo *n* will turn the array into a circle. This is still the conceptually simplest way to construct a queue in a high-level language, but it does admittedly slow things down a little, because the array indices must be compared to zero and the array size, which is comparable to the time taken to check whether an array index is out of bounds, which some languages do, but this will certainly be the method of choice for a quick and dirty implementation, or for any high-level language that does not have pointer syntax. The array size must be declared ahead of time, but some implementations simply double the declared array size when overflow occurs. Most modern languages with objects or pointers can implement or come with libraries for dynamic lists. Such data structures may have not specified a fixed capacity limit besides memory constraints. Queue overflow results from trying to add an element onto a full queue and queue underflow happens when trying to remove an element from an empty queue. A *bounded queue* is a queue limited to a fixed number of items. There are several efficient implementations of FIFO queues. An efficient implementation is one that can perform the operations—enqueuing

OSMOSE
HEATING
CONCERNS
ERGONOMIC
SEMIOETHICS
PIGMENTATION
ORGAN-SPECIFIC
MACRONUTRIENTS
CONSEQUENTIALISM

MENDES

CAUTION

ENCODING

SUBMARINE

HALORUBRUM

OPTIMIZATION

METHYLENIMINE

CHEMINFORMATIC

EPHEMERALIZATION

Cheetah
Musician
Semantics
Inflectional
Consolidation
Photosynthetic
State Integrated
Mechanosynthesis
Cultural Materialism

Minimal

Germany

Emulation

Confronting

Pseudomonas

Supercontinent

Nanoengineering

Organic Chemistry

Microbotryomycetes

In computer science, a queue is a collection of entities that are maintained in a _sequence_ and can be modified by the addition of entities at one end of the sequence and the removal of entities from the other end of the sequence. By _convention, the end of the sequence at which elements are added is called the back, tail, or rear of the queue, and the end at which_

The operation of adding an element to the rear of the queue is known as _enqueue,_ and the operation of removing an element from the front is known as _dequeue_. Other operations may also be allowed, often including a peek or front operation that returns the value of the next element to _be dequeued without dequeuing it. The operations of a queue make it a first in first out (FIFO) data structure. In a FIFO data structure, the first element added to the queue will be the first one to be removed._

In computer science, a queue is a collection of entities that are maintained in a sequence and can be modified by the addition of entities at one end of the sequence and the removal of entities from the other end of the ***sequence.*** By convention, the end of the sequence at which elements are added is called the back, tail, or rear of the queue, and the end at which *elements* are removed is called the head or front of the queue, analogously to the words used when people line up to wait for goods or services. The operation of adding an element to the rear of the queue is known as enqueue, and the operation of removing an element from the front is known as dequeue. Other operations may also be allowed, often including a peek or front operation that returns the value

In computer science, a queue is a collection of entities that are maintained in a sequence and can be modified by the addition of entities at one end of the sequence and the removal of entities from the other end of the ***sequence.*** By convention, the end of the sequence at which elements are added is called the back, tail, or rear of the queue, and the end at which ***elements*** are removed is called the head or front of the queue, analogously to the words used when people line up to wait for goods or services. The operation of adding an element **to the rear of the queue is known as enqueue, and the operation of removing an element from the front is known as dequeue. Other operations may also be allowed, often including a peek or front operation that returns**

In computer science, a queue is a collection of entities that are maintained in a sequence and can be modified by the addition of entities at one end of the sequence and the removal of entities from the other end of the ***sequence.*** By convention, the end of the sequence at which elements are added is called the back, tail, or rear of the queue, and the end at which ***elements*** are removed is called the head or front of the queue, analogously to the words used when people line up to wait for goods or services. The operation of adding an **element to the rear of the queue is known as enqueue, and the operation of removing an element from the front is known as dequeue. Other operations may also be allowed, often including a peek or front**

In computer science, a queue is a collection of entities that are maintained in a sequence and can be modified by the addition of entities at one end of the sequence and the removal of entities from the other end of the ***sequence.*** By convention, the end of the sequence at which elements are added is called the back, tail, or rear of the queue, and the end at which ***elements*** are removed is called the head or front of the queue, analogously to the words used when people line up to wait for goods or services. The operation of **adding an element to the rear of the queue is known as enqueue, and the operation of removing an element from the front is known as dequeue. Other operations may also be allowed, often including a**

OpenType Features

TABULAR FIGURES	Pi 3.1415 Pi 3.1415
FRACTIONS	5 1/16 miles 5 $^{1}/_{16}$ miles
SUPERSCRIPT	Smith3 Smith^3
SUBSCRIPT	H2O H_2O
UPPERCASE FORMS	@TYPESUPPLY @TYPESUPPLY

Character Set

UPPERCASE	ABCDEFGHIJKLMNOPQRSTUVWXYZ ÆŒØĐÞĦŁŊŦÀÁÂÃÄÅĀĂĄÇĆĊČĎĐÈÉÊËĒĔĖĘĚẼ ĞĠĢĦÌÍÎÏĨĪĬĮİĶĹĻĽĿŁÑŃŅŇŊÒÓÔÕÖŌŎŐŔŖŘ ŚŞŠȘŤŦÙÚÛÜŨŪŬŮŰŲŴẀẂẄÝŶỲŸỸŹŻŽǼIJ
LOWERCASE	abcdefgghijklmnopqrstuvwxyz ßæœøðđþħłŋŧàáâãäåāăąçćċčďđèéêëēĕėęěẽğġ ǵğġģħìíîïĩīĭįķĺļľŀłñńņňŋòóôõöōŏőŕŗřśşšșťŧùú ûüũūŭůűųŵẁẃẅýÿŷỳỹźżžǽ
FIGURES	0123456789
TABULAR	0123456789
SUPERSCRIPT & SUBSCRIPT	^{0123456789} $_{0123456789}$
FRACTIONS	$^{0123456789}/_{0123456789}$
FIGURE STUFF	$¢£¥₩€ƒ +−±×÷=≠≈<>≤≥ °#%‰
TABULAR	$¢£¥₩€ƒ
PUNCTUATION	.,:;…!¡?¿ -–—·•_'"“”‘’‚„«»‹› /\\|¦()[]{}
UPPERCASE	¡¿ /\\|¦()[]{}
SYMBOLS & REFERENCE MARKS	&@ªº *†‡~^ ¶§©℗®™
UPPERCASE	@ª
ARROWS	←→↑↓↖↗↘↙

Supported Languages

Afrikaans, Albanian, Asturian, Basque, Bosnian, Breton, Catalan, Cornish, Croation, Czech, Danish, Dutch, English, Estonian, Faroese, Finnish, French, *Galician, German, Greenlandic, Guaraní, Hawaiian, Hungarian, Icelandic, Indonesian, Irish Gaelic, Italian, Kurdish, Latin, Latvian, Lithuanian, Luxembourgish, Malagasy, Maltese, Maori, Norwegian, Occitan, Polish, Portuguese, Romanian, Romansh, Sami, Samoan, Scots, Scottish Gaelic, Slovak, Slovene, Spanish, Swahilli, Swedish, Tagalog, Turkish, Walloon, Welsh and Wolof.*

In Closing

CONTACT	*Type Supply* *122 Overbrook Rd.* *Baltimore, MD 21212* *United States* *info@typesupply.com* *typesupply.com*
LEGAL STUFF	*©2020 Type Supply LLC All rights reserved.* *Type Supply is a trademark of Type Supply LLC.* *Marigny is a trademark of Type Supply LLC.*
TEXT	*Great Cities of the United States* *by Gertrude Van Duyn Southworth and Stephen Elliott Kramer* *via gutenberg.org*