# Generic linked lists

Chetan Karwa

Arjen Markus

Milan Curcic

GSoC project

# Linked lists as a general container

- Linked lists are a well-known dynamic (abstract) data structure
- Grow and shrink
- Well-established API
- Each “node” stores data

# Goals

- Efficient implementation – useable with large amount of data (one million nodes and more)
- Flexible – store arbitrary data
- Useable as a building block:
  - Stacks
  - Queues and priority queues
  - Associative arrays
  - ...

# Performance

- Sequential access -> locating an item is O(N)
- Solution for long lists: simple hierarchical structure
    - One “parent list” whose data are “child lists”
    - Quickly get the right child list – O(√N), roughly
- Layered implementation
- Some timings: length 1 million nodes

| | |
|---|---|
| - built: | 0.734375 s |
| - traverse: | 0.015620 s |
| - delete: | 0.093750 s |

# Examples of usage: fill and retrieve (1)

```
use Linked_List
integer           :: i
type(Parent_List) :: List
character(len = 1000) :: mystr
class(*), pointer :: data

mystr = repeat( 'a', 1000 )
call List%append( mystr )
call List%append( 321 )
call List%append( 'Hello' )
```

# Examples of usage: fill and retrieve (2)

```
…
data => List%get(3)
select type (v => data)
    type is (character(len=*))
        write(*,*) v
end select
```

# Examples of usage: priority queue (1)

```
use prio
implicit none

type(prio_queue)          :: queue
type(prio_item)           :: item
type(prio_item), pointer :: item_p

!
! Fill the list with arbitrary data - not sorted
!
item = prio_item( 70, 'Task 1') ; call queue%push_elem( item )
item = prio_item( 40, 'Task 2') ; call queue%push_elem( item )
```

# Examples of usage: priority queue (2)

```
…
do
    call queue%get_and_remove( item_p )
    if ( .not. associated( item_p ) ) exit

    select type ( value => item_p%data )
        type is ( character(len=*) )
            write(*,*) item_p%priority, value
    end select
enddo
```

# Future work

- Extend the API

  For instance: traverse the list with a callback routine

- Systematic test programs (stdlib)
- Performance tests
- User documentation