FCL-STL, a Generics-Based Template Library for FreePascal

Vladimír Boža[1], Michal Forišek[1]

[1]Comenius University, Bratislava, Slovakia

IOI 2012, Sirmione, Italy

Generic programming

The problem

- Everyone needs to sort, search, enumerate, . . .
- Everyone's data is different.
- We don't want to re-implement the same stuff 100 times. (A function that sorts ints. Another one for longs.)

The solution (for strongly typed languages)

Introduce metavariables that represent types:

```
void swap(T &a, T &b) { T c=a; a=b; b=c; }
```

Write every algorithm / data structure once.

Generic programming

The problem

- Everyone needs to sort, search, enumerate, . . .
- Everyone's data is different.
- We don't want to re-implement the same stuff 100 times. (A function that sorts ints. Another one for longs.)

The solution (for strongly typed languages)

Introduce metavariables that represent types:

```
void swap(T &a, T &b) { T c=a; a=b; b=c; }
```

Write every algorithm / data structure once.

C++ vs. Pascal on IOI

C++

```
sort(A, A+n);
```

Pascal

```
procedure Sortrange(var Arr:TArr; Start,Finish:SizeUInt);
var pivot,temp:TValue; i,j:SizeUInt;
begin
  pivot:=Arr[Start]; i:=Start-1; j:=Finish;
  repeat
    repeat
      dec(j);
    until (not (pivot < Arr[j]));
    repeat
      inc(i);
    until (not (Arr[i] < pivot));
    if(i < j) then
    begin
      temp:=Arr[i]; Arr[i]:=Arr[j]; Arr[j]:=temp;
    end;
  until (i>=j);
  Sortrange(Arr, Start, j+1);
  Sortrange(Arr, j+1, Finish);
end;
```

Content

FreePascal FCL-STL	C++ STL equivalent
Sort	sort
RandomShuffle	random_shuffle
NextPermutation	next_permutation
---	stable_sort, nth_element
TVector	vector
TStack, TQueue	stack, queue
TDeque	deque
TPriorityQueue	priority_queue
TSet, Tmap	set, map
THashSet, THashMap	unordered_set, unordered_map
---	list
---	bitset
---	(unordered) multiset

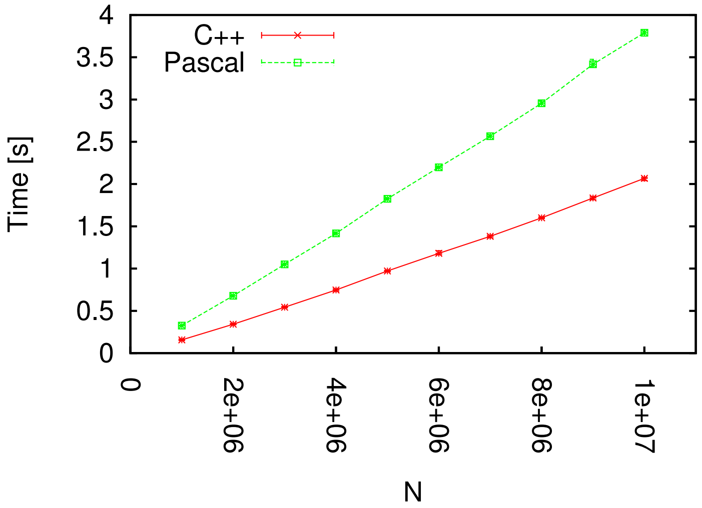
Usage example - random shuffle + sort

```
uses garrayutils, gutil, gvector;

type iLess = specialize TLess<longint>;
     iVector = specialize TVector<longint>;
     iOrdUtils = specialize TOrderingArrayUtils<
         iVector, longint, iLess>;
     iUtils = specialize TArrayUtils<iVector, longint>

var V : iVector; n, i : longint;

begin
  read(n);
  V := iVector.Create;
  for i := 0 to n-1 do V.PushBack(i);
  iUtils.RandomShuffle(V, n);
  iOrdUtils.Sort(V, n);
end.
```

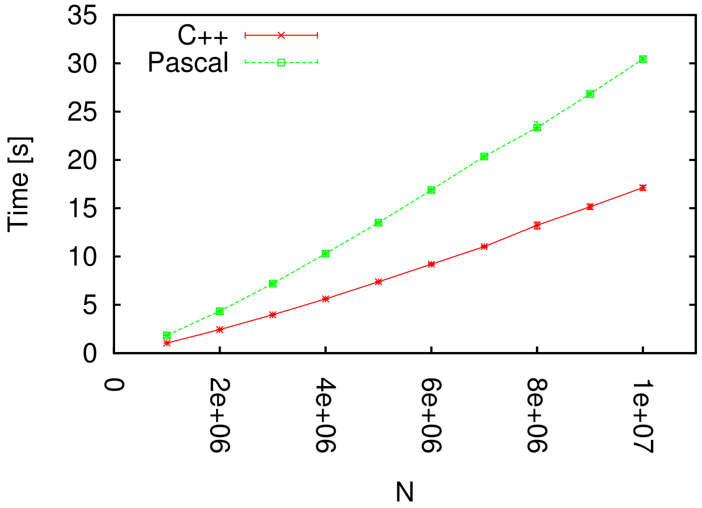
Usage example - sets and set iterators

```
uses gvector, gset, gutil;

type iLess   = specialize TLess<longint>;
     iVector = specialize TVector<longint>;
     iSet    = specialize TSet<longint,iLess>;

var V : iVector; S : iSet; N, i : longint; it : iSet.TIterator;

begin
    read(N);
    V := iVector.Create();
    for i:=1 to N do V.PushBack(random(N));
    S := iSet.Create();
    for i:=0 to N-1 do S.Insert(V[i]);
    V.Clear(); it := S.Min();
    repeat V.PushBack( it.GetData() ); until not it.Next();
end.
```

Performance - sorting



Performance - sets and iterators



Current status

- Not in current stable branch of FreePascal (2.6.0).
- Should be in part of next stable release (2.6.x).
- Already in the development branch (2.7.x) for some months.

Thank you for your attention!