Lập trình tổng quát (Generic programming)

Created by anhtt-fit@mail.hut.edu.vn
Edited by huonglt-fit@mail.hut.edu.vn

Giới thiệu

- Lập trình tổng quát là phương pháp tạo ra các đoạn chương trình tổng quát cho phép dễ dàng sử dụng lại trong nhiều bài toán khác nhau.
- Một ví dụ đơn giản của lập trình tổng quát là hàm memcpy() function trong thư viện chuẩn của C. Đây là hàm tổng quát cho phép chép dữ liệu từ 1 vùng lưu trữ sang 1 nơi khác.
  - void* memcpy(void* region1, const void* region2, size_t n);
- Hàm memcpy() đã được tổng quát hóa bằng cách sử dụng cách khai báo void*. Cách khai báo này cho phép hàm có thể sử dụng để sao chép nhiều kiểu dữ liệu khác nhau.
- 1 cách tổng quát, để sao chép dữ liệu, ta chỉ cần biết địa chỉ và kích cỡ của vùng cần sao chép.

memcpy

- 1 cài đặt của hàm memcpy():

```
void* memcpy(void* region1,
             const void* region2,
             size_t n) {
   const char* first = (const char*) region2;
   const char* last = ((const char*) region2) + n;
   char* result = (char*) region1;
   while (first != last) *result++ = *first++;
   return result;
}
```

Các hàm tổng quát

- Trong hàm tổng quát, dữ liệu cần truyền theo cách tổng quát (qua địa chỉ và kích cỡ)
- Nếu thuật toán yêu cầu một hàm cụ thể để thao tác với dữ liệu cụ thể (như so sánh 2 giá trị), hàm tổng quát cần được truyền qua con trỏ hàm.
- Ví dụ: Hàm tìm kiếm tổng quát trên mảng
  - Làm cách nào để truyền dữ liệu cho hàm?
  - Làm thế nào để hàm có thể kiểm tra 2 dữ liệu trong mảng có bằng nhau hay không?

Cài đặt (1)

- Một hàm tổng quát cần được truyền qua các tham số sau:
  - void * buf: địa chỉ của buffer chứa dữ liệu của mảng
  - int size: kích thước 1 phần tử trong mảng
  - int total: số các phần tử của mảng
- Thuật toán tìm kiếm cần 1 hàm để so sánh các phần tử trong mảng. Các phần tử này được truyền co hàm so sánh thông qua địa chỉ của nó. Sử dụng hàm con trỏ để biểu diễn 1 hàm so sánh tổng quát.
  - int (*compare) (void * item1, void * item2)

Cài đặt (2)

```
// return -1 if not found
int search(  void* buf,
             int size,
             int l, int r,
             void * item,          const    const
            int (*compare)(void*, void*)) {
  if (r < l) return -1;
  i = (l + r)/2;
  res = compare( item, (char*)buf+(size*i) );
  if (res==0)
      return i;
  else if (res < 0)
      return search(buf, size, l, i-1, item, compare);
  else
      return search(buf, size, i+1, r, item, compare);
}
```

Cách sử dụng

```
int int_compare(void const* x, void const *y) {
   int m, n;
   m = *((int*)x);
   n = *((int*)y);
   if ( m == n ) return 0;
   return m > n ? 1: -1;
}
int main() {
   int a[100];
   int n = 100, item = 5;
   for (i=0; i<n; i++) a[i] = rand();
   qsort(a, n, sizeof(int), int_compare);
   res = search (a, sizeof(int), 0, n-1, int_compare);
}
```

Bài tập 1

- Viết thuật toán sắp xếp tổng quát của riêng bạn sử dụng thuật toán đã giới thiệu trong bài giảng 1.
- Viết lại các chương trình của bạn trong bài giảng 1 sử dụng hàm tổng quát.

Hướng dẫn

- Để đổi chỗ 2 phần tử trong mảng, ta cần xây dựng 1 hàm đổi chỗ tổng quát như sau
  - void exch (void * buf, int size, int i, int j);

Kiểu dữ liệu tổng quát

- Làm cách nào để tạo 1 vùng lưu dữ liệu tổng quát, với dữ liệu có thể là kiểu integer, float, char, bản ghi…?
- Kiểu dữ liệu tổng quát phải có lợi để xây dựng các ADT tổng quát trong C như linked list, binary tree, …
- Kiểu tập hợp có thể là 1 cách hay để xây dựng kiểu dữ liệu tổng quát

Jval (libfdr lib)

```
typedef union {
    int i;
    long l;
    float f;
    double d;
    void *v;
    char *s;
    char c;
} Jval;
```

- Jval có thể dùng để lưu nhiều kiểu dữ liệu khác nhau. Ví dụ:

```
Jval a, b;
a.i = 5;
b.f = 3.14;
```

Hàm thiết lập (Constructor functions)

- Để đơn giản hóa cách sử dụng Jval, một số hàm thiết lập được xây dựng
  - Jval new_jval_i(int);
  - Jval new_jval_f(float);
  - Jval new_jval_d(double);
  - Jval new_jval_s(char *);
- Example:

```
Jval a, b;
a = new_jval_i(5);
b = new_jval_f(3.14);
```

Hàm truy cập (Access functions)

- Để đọc dữ liệu từ hàm tổng quát, các hàm truy cập cần được sử dụng cho các kiểu dữ liệu cụ thể:
  - int jval_i(Jval);
  - float jval_f(Jval);
  - double jval_d(Jval);
  - char* jval_s(Jval);
- Ví dụ:

```
Jval a, b;
a = new_jval_i(5);
b = new_jval_float(3.14);
printf("%d", jval_i(a));
printf("%f", jval_f(a));
```

Bài tập 2

- Viết lại hàm sắp xếp và tìm kiếm tổng quát sử dụng Jval để biểu diễn các vùng lưu dữ liệu như sau:
  - void sort_gen (Jval a[], int l, int r, int (*compare)(Jval*, Jval*));
  - int search_gen (Jval a[], int l, int r, Jval item, int (*compare)(Jval*, Jval*));

Hướng dẫn

- Sau khi tạo hàm sắp xếp và tìm kiếm tổng quát, bạn có thể tạo các hàm để thao tác trên các kiểu dữ liệu như sau:

```
int compare_i(Jval* a, Jval* b);
void sort_i (Jval a[], int l, int r);
int search_i (Jval a[], int l, int r, int x);
Jval* create_array_i (int n);
```