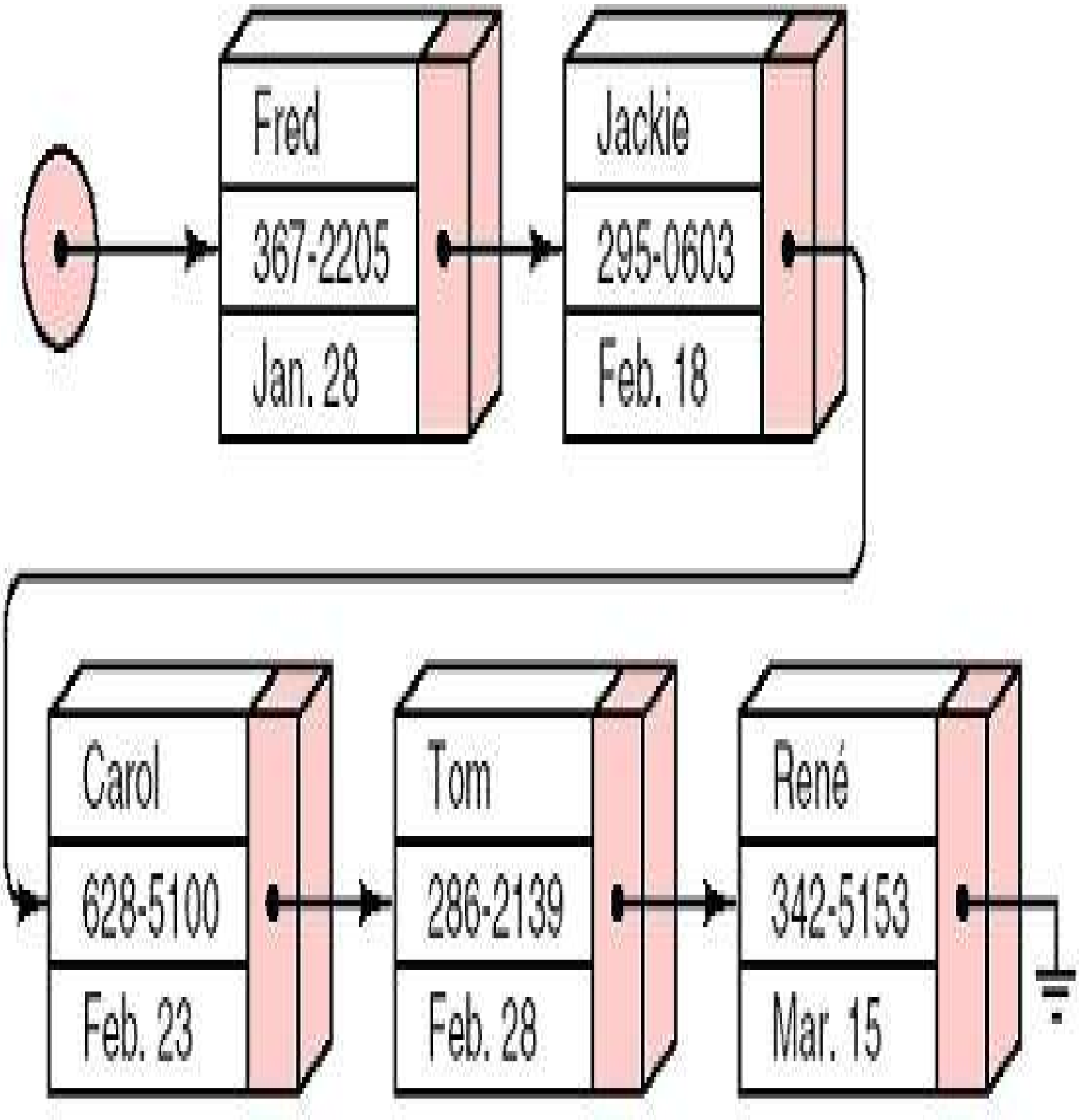

Fred
367-2205
Jan. 28
Jackie
295-0603
Feb. 18
Carol
628-5100
Feb. 23
Tom
286-2139
Feb. 28
René
342-5153
Mar. 15

BỘ NHỚ ĐỘNG TRONG C++

batanlp

Chào mừng các bạn đón đọc đầu sách từ dự án sách cho thiết bị di động
Nguồn: http://vnthuquan.net
Phát hành: Nguyễn Kim Vỹ.

Mục lục

batanlp

BỘ NHỚ ĐỘNG TRONG C++

Trong bài viết này, tôi sẽ giới thiệu với các bạn về cách lập trình với con trỏ (**pointer**) trong việc cấp phát bộ nhớ động (**dynamic memory alocation**) bằng ngôn ngữ C++. Đối với các bạn đã có kinh nghiệm lập trình với C++ thì bài này đối với các bạn chỉ là "a-bờ-cờ" mà thôi. Nhưng với các bạn mới học C++ thì có lẽ là bổ ích.

Nếu bạn mới học lập trình và làm quen với cấu trúc dữ liệu thì một trong số những cấu trúc dữ liệu đầu tiên mà bạn "rớ" tới là **Stack** (ngăn xếp) và **Queue** (hàng đợi). Bạn có thể dùng 1 mảng (**array**) để thiết kế Stack và Queue. Dùng mảng thì đơn giản, nhưng bạn sẽ gặp một số bất lợi như sau:

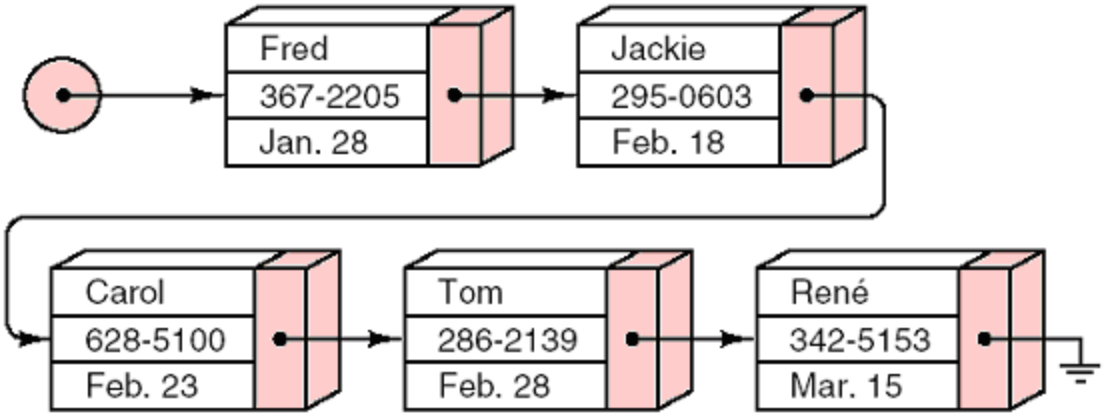
- **Lãng phí bộ nhớ**: giả sử bạn khai báo **int array_entry[100]** thì bạn sẽ có 1 vùng nhớ cho 100 phần tử, nhưng nếu chương trình của bạn chỉ thường xuyên dùng có 10 phần tử, và 1 vài lần là dùng đến 100 phần tử thì tức là bạn đã phí phạm 90 phần tử.
- **Thiếu bộ nhớ**: vì tiết kiệm, bạn chỉ khai báo **int array_entry[10]**, nhưng nếu bạn cần dùng đến 15 phần tử thì...overflow ngay. Chưa hết, array cần một vùng nhớ liên tục, giả sử máy bạn vẫn còn nhiều bộ nhớ trống, nhưng không có vùng nhớ trống liên tục nào đủ lớn cho mảng của bạn. Thế là vẫn...thiếu bộ nhớ.
- Và cuối cùng, người ta khi thấy bạn viết như vậy thì coi bạn là dân amateur, buồn nhỉ? L

Nhưng không sao, bài viết này sẽ giúp bạn vượt qua các trở ngại đó. Bạn sẽ

biết cách ứng dụng con trỏ để tạo thành các cấu trúc dữ liệu kiên kết (**linked structure**) để tạo thành các **linked stack** hoặc **linked queue**. Bạn sẽ biết được cách toàn quyền cấp phát bộ nhớ khi có nhu cầu sử dụng; và khi không dùng nữa thì ta xoá nó đi, nhường memory cho các chương trình khác.

Ghi chú: xem như là bạn đã có học và biết qua sơ sơ về C++ rồi, nhứng đoạn code trong bài viết này là C++ chứ không phải là C.

Ứng dụng con trỏ, ta có thể tạo ra một danh sách liên kết (mảng liên kết - **linked list**) như sau:



Phần đánh dấu màu hồng (và nút tròn đầu tiên) là các con trỏ, có nhiệm vụ link đến phần tử tiếp theo trong danh sách. Nhờ có các link này mà khi đang ở phần tử thứ nhất, bạn sẽ có manh mối để lần đến phần tử thứ hai, rồi phần tử thứ ba...

Một phần tử của danh sách có thể được biểu diễn qua cấu trúc như sau (1 phần tử ta gọi là 1 Node):

```
struct Node {
   // data members
   Node_entry entry;
   Node *next;
   //constructors
   Node();
```

```
    Node(Node_entry item, Node *add_on = NULL);
};
```

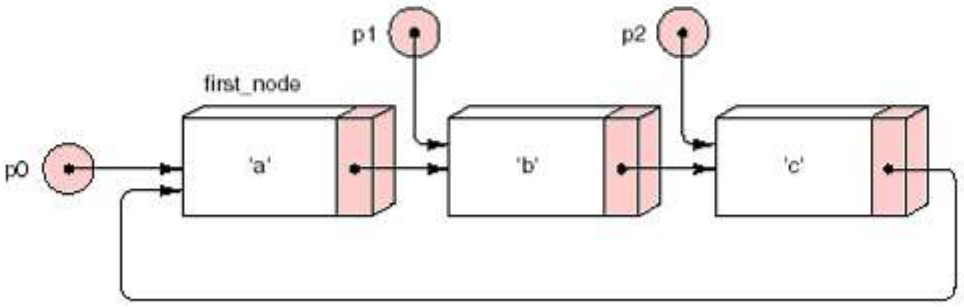
Ghi chú:**Node_entry** là kiểu dữ liệu để chứa data của phần tử, entry chính là nơi data của phần tử được lưu. Ở ví dụ trong hình trên, với node đầu tiên thì entry sẽ chứa **Fred**, **367-2205** và **Jan. 28**.

Và ta có các hàm khởi tạo cho nó như sau:

```
Node::Node()
{
   next = NULL;
}
Node::Node(Node_entry item, Node *add_on)
{
   entry = item;
   next = add_on;
}
```

Như vậy, khi ta khai báo **Node* data** và khởi tạo xong cho **data** thì **data->entry** sẽ là phần dữ liệu của node đó, và **data->next** sẽ là đường dẫn tới node tiếp theo. Như thế thì ta có thể duyệt hết toàn bộ data mà ta đã lưu trong danh sách rồi. Để rõ hơn, ta hãy xem một ví dụ qua đoạn code nhỏ sau. Chú ý: khi **data->next == NULL** thì tức là node này là phần tử cuối cùng.

```
Node first_node('a');     //  Node first_node stores data 'a'.
Node *p0 =  &first_node;  //  p0 points to first_Node.
Node *p1 = new Node('b');  //  A second node storing 'b' is created.
p0->next = p1;            //  The second Node is linked after first_node.
Node *p2 = new Node('c', p0);  //  A third Node storing 'c' is created.
   //  The third Node links back to the first node, *p0.
p1->next = p2;            //  The third Node is linked after the second Node.
```



Đoạn code trong ví dụ sẽ tạo ra một danh sách liên kết như trong hình trên (không tin thì bạn...thử xem là biết liền J). Địa chỉ bộ nhớ của các node thì hoàn toàn ngẫu nhiên và không liên tục vì nó là **dynamic memory allocation** mà.

Thế là đã Ok cho việc tạo một cấu trúc liên kết. Giờ ta bắt tay vào xây dựng **Linked Stack** và **Linked Queue**. Tôi sẽ trình bày phần **Linked Stack** thôi, phần **Linked Queue** thì các bạn hãy phát triển thêm nhé. Cũng hoàn toàn tương tự thôi.

Ta khai báo Stack như sau:

```
enum Error_code {underflow, success, overflow};
typedef int Stack_entry;
class Stack {
public:
   Stack();
   Bool empty() const;
   Stack_entry push(const Stack_entry &item);
   Stack_entry top(Stack_entry &item) const;
   Error_code pop();
protected:
```

```
    Node *top_node;
}
```

Sau đó là các đoạn code cho các hàm trong Stack như sau:

```
Stack_entry Stack::push(const Stack_entry &item)
{
    Node *new_top = new Node(item, top_node);
    if (new_top == NULL) return 0;
    top_node = new_top;
    return item;
}
/**********************************************************************
************/
Error_code Stack::pop()
{
    Node *old_top = top_node;
    if (top_node == NULL) return underflow;
    top_node = old_top->next ;
    delete old_top;
    count--;
    return success;
}
/**********************************************************************
************/
Stack_entry Stack::top(Stack_entry &item) const
{
    if (top_node == NULL) return 0;
    item = top_node->entry;
    return item;
}
/**********************************************************************
```

```
***********/
bool Stack::empty() const
{
   if (top_node == NULL) return true;
   return false;
}
/*************************************************************
***********/
Stack::Stack()
{
   top_node = NULL;
   count = 0;
}
```

Như thế đó, ta đã có 1 **Linked Stack** mà không dùng array. Dữ liệu được nhập vào tới đâu thì bộ nhớ được cấp phát tới đó. Và khi dữ liệu được lấy ra khỏi stack thì phần bộ nhớ của nó được giải phóng ngay lập tức, để dành cho việc khác.

Các bạn tự nghiên cứu phần Queue nhé! Nếu bạn muốn ngâm cứu bản gốc (tiếng Anh) thì bạn có thể email cho tôi: batanlp@hotmail.com.

Lời cuối: Cám ơn bạn đã theo dõi hết cuốn truyện.
Nguồn: http://vnthuquan.net
Phát hành: Nguyễn Kim Vỹ.

Nguồn: www.diendantinhoc.net
Được bạn: mickey đưa lên
vào ngày: 19 tháng 8 năm 2004