**Bài tập**

Cho mảng các số nguyên n phần tử được nhập từ bàn phím (mảng này được cấp phát động và quản lý thông qua con trỏ arr), bạn hãy viết chương trình tính tổng của các phần tử trong mảng và hiển thị ra màn hình:

Sum = {P}

Với {P} là tổng của các phần tử trong mảng.

Nếu bạn chưa biết về khái niệm cấp phát bộ nhớ động thì có thể xem phần lý thuyết bên dưới.

**Lý thuyết**

Trước hết bạn hãy xem lại cách khai báo mảng thông thường:

// Khai báo mảng a 100 phần tử

int arr[100];

**Nhược điểm:**

* Với cách khai báo này số phần tử của mảng luôn phải cố định, bạn không thể thay đổi kích thước của mảng khi chương trình đang chạy. Ví dụ vì lý do nào đó mà sau này mảng arr cần tới 101 phần tử thì bạn chỉ có cách tắt chương trình đi và code lại thành int arr[101]; (do mảng không tự tăng kích thước được).
* Mảng sẽ được cấp phát trên vùng nhớ Stack, và kích thước của vùng nhớ này là nhỏ nên bạn không thể khai báo mảng có nhiều phần tử được. Ví dụ bạn hãy thử chạy chương trình sau:
* #include <iostream>
* using namespace std;
* int main() {
* int arr[10000000];
* return 0;
* }

​

Chương trình này sẽ báo lỗi do kích thước của vùng nhớ Stack không đủ để cấp phát bộ nhớ cho mảng a.

**Ưu điểm:**

* Mảng được cấp phát trên vùng nhớ Stack nên sẽ có tốc độ truy xuất nhanh (vùng nhớ stack tuy có kích thước nhỏ nhưng lại có tốc độ truy xuất nhanh hơn các vùng nhớ khác).
* Bộ nhớ được chương trình quản lý tự động (vùng nhớ được cấp phát cho mảng sẽ được chương trình giải phóng khi không dùng nữa).

Mảng được cấp phát động sẽ giải quyết được những **nhược điểm** trên nhưng cũng có một số nhược điểm. Trước hết bạn hãy xem cách để cấp phát bộ nhớ động cho mảng:

// Cấp phát bộ nhớ động cho mảng có 10000000 phần tử

// Để cấp phát bộ nhớ động cho mảng ta sử dụng toán tử new

int\* arr = new int[10000000];

Đoạn code trên cấp phát bộ nhớ động cho mảng có 10000000 phần tử, lưu ý rằng arr không phải là một mảng, arr là một con trỏ trỏ tới vùng nhớ được cấp phát động (trỏ tới phần tử đầu tiên của mảng được cấp phát động) hay hiểu cách khác arr là con trỏ dùng để quản lý mảng có 10000000 phần tử.

Bạn có thể sử dụng mảng được cấp phát động giống như mảng thông thường:

#include <iostream>

using namespace std;

int main() {

// Cấp phát động cho mảng có 10 phần tử

int\* arr = new int[10];

for (int i = 0; i < 10; i++) {

arr[i] = i;

}

for (int i = 0; i < 10; i++) {

cout << arr[i] << " ";

}

// Giải phóng bộ nhớ được cấp phát động khi không sử dụng

delete[]arr;

return 0;

}

Kết quả khi chạy chương trình:

0 1 2 3 4 5 6 7 8 9

**Ưu điểm của cấp phát bộ nhớ động**

* Số phần tử của mảng không nhất thiết phải cố định. Ví dụ:
* #include <iostream>
* using namespace std;
* int main() {
* int n;
* cin >> n;
* // Cấp phát động cho mảng có n phần tử
* int\* arr = new int[n];
* // Giải phóng bộ nhớ được cấp phát động khi không sử dụng
* delete[]arr;
* return 0;
* }

​​

Có thể thấy trong ví dụ trên số phần tử của mảng được cấp phát động là một biến được nhập từ bàn phím.

* Bạn có thể thay đổi kích thước mảng khi chương trình đang chạy (bản chất là cho con trỏ tới vùng nhớ khác, ví dụ nếu bạn cần tăng kích thước của mảng thì bạn chỉ cần cho con trỏ trỏ tới vùng nhớ có kích thước lớn hơn).
* Mảng được cấp phát trên vùng nhớ Heap mà vùng nhớ Heap là vùng nhớ có kích thước lớn nhất trong các vùng nhớ nên với cấp phát động bạn có thể khai báo mảng với nhiều phần tử. Ví dụ:
* #include <iostream>
* using namespace std;
* int main() {
* int\* arr = new int[10000000];
* // Giải phóng bộ nhớ được cấp phát động khi không sử dụng
* delete[]arr;
* return 0;
* }

​

Chương trình này khi chạy sẽ không báo lỗi do vùng nhớ Heap có kích thước lớn và có thể lưu trữ được mảng có 10000000 phần tử.

**Nhược điểm của cấp phát bộ nhớ động**

* Vùng nhớ Heap có tốc độ truy xuất chậm hơn vùng nhớ Stack nên việc truy xuất vào các phần tử trong mảng được cấp phát động sẽ chậm hơn.
* Bạn phải tự quản lý bộ nhớ được cấp phát động (chương trình sẽ không quản lý cho bạn). Nếu bạn thực hiện cấp phát bộ nhớ động mà không giải phóng thì vùng nhớ này sẽ vẫn còn cho tới khi khởi động lại máy (gây lãng phí bộ nhớ, tệ hơn nữa là tràn bộ nhớ).

Để giải phóng bộ nhớ được cấp phát động bạn dùng toán tử delete[]. Ví dụ:

#include <iostream>

using namespace std;

int main() {

// Cấp phát bộ nhớ động cho mảng có 1000 phần tử

int\* arr = new int[1000];

...

// Giải phòng bộ nhớ khi không dùng tới

delete[] arr;

return 0;

}

​​

Do đó khi cấp phát động hãy luôn nhớ phải dùng toán tử delete[] để giải phóng bộ nhớ khi không dùng tới nữa.

Lỗi thường gặp khi cấp phát bộ nhớ động:

#include<iostream>

using namespace std;

int main() {

int\* arr = new int[1000];

arr = new int[2000];

delete[]arr;

return 0;

}

Đoạn code trên sẽ gây ra rò rỉ bộ nhớ do khi gán arr = new int[2000]; thì vùng nhớ được cấp phát động cho mảng có 1000 phần tử vẫn còn đó, bạn cần giải phóng vùng nhớ này trước khi cho arr trỏ tới vùng nhớ mới giống như sau:

#include<iostream>

using namespace std;

int main() {

int\* arr = new int[1000];

delete[]arr;

arr = new int[2000];

delete[]arr;

return 0;

}

Đọc tới đây bạn đã hiểu về cấp phát bộ nhớ động và biết cách sử dụng mảng được cấp phát bộ nhớ động, hãy quay lại phần bài tập và làm thử.

Nếu bạn chưa làm được bài này thì có thể xem hướng dẫn ở bên dưới.

**Hướng dẫn**

Code mẫu:

#include <iostream>

using namespace std;

int main() {

int n;

int\* arr;

cin >> n;

// Thực hiện cấp phát bộ nhớ động cho mảng

arr = new int[n];

// Nhập dữ liệu cho mảng

for (int i = 0; i < n; i++) {

cin >> arr[i];

}

// Tính tổng các phần tử trong mảng và lưu vào biến sum

int sum = 0;

for (int i = 0; i < n; i++) {

sum += arr[i];

}

cout << "Sum = " << sum;

delete[]arr;

return 0;

}