```
/* Chuong trinh tinh sin(x) theo cong thuc:
sin(x) = x - x^3/3! + x^5/5! + ... + (-1)^n*x^(2n+1)/(2n+1)! +
...  voi sai so cho truoc                                    */
#include <stdio.h>
#include <conio.h>
#include <math.h>
void main()
{
  float sinx,sh,n,x,saiso;
  clrscr();
  printf("Nhap x = ");
  scanf("%f",&x);
  printf("Nhap sai so = ");
  scanf("%f",&saiso);
  n=0.0;
  sinx=x;
  sh=x;
  do
  {
    n += 1.0;
    sh *= -x*x/((2*n)*(2*n+1));
    sinx += sh;
  } while (fabs(sh) >= saiso);
  printf("\nsin(%f) = %f",x,sinx);
  printf("\nsin(%f) cua may tinh = %f",x,sin(x));
  getch();
}
```

CHƯƠNG 3: CON TRỎ VÀ CẤP PHÁT BỘ NHỚ ĐỘNG

3.1. Con trỏ

3.1.1. Khái niệm và cách khai báo con trỏ

Con trỏ là một biến có giá trị là địa chỉ của một đối tượng khác; đối tượng ở đây có thể là một biến hoặc một hàm, mà không thể là một hằng.

Khai báo biến con trỏ:

< kiểu > *< danh sách biến trỏ > ;

Ví dụ:

```
int *pi, *qi ;       //  khai báo 2 con trỏ kiểu int
float *pf, *qf ;     //  khai báo 2 con trỏ kiểu float
char *pc , *qc ;     //  khai báo 2 con trỏ kiểu char
```

CHƯƠNG 3: CON TRỎ VÀ CẤP PHÁT BỘ NHỚ ĐỘNG

3.1.2. Toán tử & và *

Xét các khai báo biến sau:

```
int   m , n , p , *pi ;
float  x , y , z ;
char  ch1 , ch2 ;
```

Toán tử & cho ta địa chỉ của một biến, còn biến con trỏ có giá trị là địa chỉ của một biến. Như vậy chỉ thị:

```
pi  =  &n ;
```

là hợp lệ và nó gán địa chỉ của biến nguyên n cho con trỏ pi. Khi đó ta nói rằng pi trỏ tới n.

Khi pi trỏ tới n, ta sẽ có 2 cách để mô tả giá trị của n:

- Hoặc bằng cách trực tiếp bằng tên n của biến như truyền thống.
- Hoặc bằng cách áp dụng toán tử * lên biến trỏ pi.

Nói cách khác, *pi tương đương với n

CHƯƠNG 3: CON TRỎ VÀ CẤP PHÁT BỘ NHỚ ĐỘNG

3.1.3. Các phép toán trên con trỏ

a/ Phép gán trên 2 con trỏ cùng kiểu

b/ Phép cộng, trừ con trỏ với số nguyên: Khi cộng hoặc trừ con trỏ với một số nguyên n, ta sẽ được một địa chỉ mới chỉ đến một biến khác nằm cách biến trước đó n vị trí.

c/ Phép trừ 2 con trỏ cùng kiểu: Cho kết quả là một số nguyên biểu thị "khoảng cách" giữa 2 biến con trỏ.

d/ Các phép quan hệ trên 2 con trỏ cùng kiểu: tiêu chuẩn so sánh dựa trên vị trí ô nhớ tương ứng với 2 con trỏ.

CHƯƠNG 3: CON TRỎ VÀ CẤP PHÁT BỘ NHỚ ĐỘNG

3.2. Liên hệ giữa con trỏ và mảng

3.2.1. Con trỏ và mảng một chiều

Ví dụ:

```
int  a[10], *pa;
```

- Phép gán `pa = &a[0]` sẽ đặt pa trỏ tới phần tử thứ nhất của mảng a, tức là pa chứa địa chỉ của a[0].
- Bây giờ phép gán x= *pa sẽ sao nội dung của a[0] vào x.
- Nếu pa trỏ tới 1 phần tử của mảng a thì theo định nghĩa pa+1 sẽ trỏ tới phần tử tiếp theo và pa + i sẽ trỏ tới phần tử thứ i sau pa. Vậy nếu pa trỏ tới a[0] thì *(pa+i) sẽ cho nội dung của a[i].
- Vì tên mảng là đồng nghĩa với địa chỉ của phần tử thứ 0 của mảng, nên phép gán pa = &a[0] cũng có thể viết là:

```
pa = a;
```

- Sau phép gán này ta có thể sử dụng pa như một tên mảng.
- Bốn cách viết sau có tác dụng như nhau: a[i] , *(a+i), *(pa+i), pa[i].

CHƯƠNG 3: CON TRỎ VÀ CẤP PHÁT BỘ NHỚ ĐỘNG

3.2.2. Con trỏ và mảng hai chiều

- Phép toán lấy địa chỉ & không dùng được đối với các phần tử của mảng nhiều chiều.
- Câu lệnh &a[i][j] là không hợp lệ và gây ra lỗi.
- Để khắc phục, ta nhập gián tiếp qua một biến trung gian như sau: Đọc 1 giá trị và chứa tạm vào một biến trung gian, sau đó ta gán biến này cho phần tử mảng.

```
for (i=0; i<m; i++)
     for (j=0; j<n; j++)
       {
             scanf("%f", &tam );
             a[i][j] = tam ;
       }
```

CHƯƠNG 3: CON TRỎ VÀ CẤP PHÁT BỘ NHỚ ĐỘNG

- Phép cộng địa chỉ trong mảng 2 chiều

Ta xét ví dụ: float a[2][3] ;

Ta đã biết *a* là địa chỉ đầu tiên của mảng.

a trỏ tới đầu hàng thứ nhất (phần tử a[0][0])

a+1 trỏ tới đầu hàng thứ hai (phần tử a[1][0])

.....

- Con trỏ và mảng 2 chiều: Để lần lượt duyệt các phần tử của mảng 2 chiều ta có thể dùng con trỏ theo cách sau:

Ví dụ: float a[2][3] , *pa;

pa = (float*) a ;

Khi đó :

pa trỏ tới a[0][0], pa+1 trỏ tới a[0][1],

pa+2 trỏ tới a[0][2], pa+3 trỏ tới a[1][0],

pa+4 trỏ tới a[1][1], pa+5 trỏ tới a[1][2]

CHƯƠNG 3: CON TRỎ VÀ CẤP PHÁT BỘ NHỚ ĐỘNG

3.2.3. Mảng các con trỏ

Khi kiểu phần tử của một mảng là kiểu con trỏ thì ta sẽ có một mảng các con trỏ.

3.2.4. Con trỏ đa cấp

Bản thân biến con trỏ cũng có địa chỉ, do đó có thể chứa địa chỉ của nó trong một đối tượng khác. Người ta gọi những đối tượng có giá trị là địa chỉ của một biến trỏ là con trỏ đa cấp. Trong khai báo biến trỏ đa cấp, ta sử dụng một số dấu "*" trước tên biến trỏ, số lượng dấu "*" xác định cấp của con trỏ.

Ví dụ:

```
int  *pi ;
int  **ppi ;
```

CHƯƠNG 3: CON TRỎ VÀ CẤP PHÁT BỘ NHỚ ĐỘNG

Ví dụ:

```
#include <stdio.h>
#include <conio.h>
void main()
{
   char *tenngay[15] = {"Thu Hai","Thu Ba","Thu Tu",
       "Thu Nam", "Thu Sau","Thu Bay","Chu Nhat"};
   char   **pp ;
   int   i;
   clrscr( );
   pp = tenngay;    // tên mảng là địa chỉ của phần tử đầu
tiên
   printf("Cac ngay trong tuan:");
   for(i=0;   i<7;  i++) printf("\n%s",pp[i]);
   getch();
}
```

CHƯƠNG 3: CON TRỎ VÀ CẤP PHÁT BỘ NHỚ ĐỘNG

3.2.5. Cấp phát động bộ nhớ

Các tệp tiêu đề liên quan: ”alloc.h”, “stdlib.h”

a/ Hàm malloc: Dạng hàm: malloc(size);

Hàm malloc dùng để cấp phát một vùng nhớ có kích thước size byte. Trong trường hợp thành công, hàm malloc trả về một con trỏ tới khối nhớ mới được cung cấp. Trong trường hợp có lỗi, hàm malloc trả về giá trị NULL.

Ví dụ:

```
// cấp phát một mảng 100 số nguyên kiểu int:
pi = (int *) malloc (100 * sizeof(int)) ;
// cấp phát một mảng 10 số thực kiểu double:
pd = (double *) malloc (10 * sizeof ( double) ) ;
// cấp phát ma trận thực kiểu double cấp 20 x 10:
ppd = (double **) malloc (20 * sizeof ( double *) ) ;
for (i=0; i < 20; i++)
      ppd[i]    =    (double   *)   malloc   (10   *   sizeof   (
double) ) ;
```

CHƯƠNG 3: CON TRỎ VÀ CẤP PHÁT BỘ NHỚ ĐỘNG

b/ Hàm calloc

Dạng hàm:

calloc (nitems , size) ;

Hàm calloc cấp phát vùng nhớ kích thước *(nitems * size)* byte và xóa trắng vùng nhớ này.

c/ Hàm free

Cú pháp:

free (< địa chỉ >) ;

Hàm free giải phóng một khối nhớ đã được cấp phát trước đó bằng hàm malloc hoặc calloc. Địa chỉ vùng nhớ được giải phóng được truyền làm tham số của hàm free.

CHƯƠNG 3: CON TRỎ VÀ CẤP PHÁT BỘ NHỚ ĐỘNG

Ví dụ: Cấp phát bộ nhớ động cho ma trận $p_{m \times n}$

```
float   **p;
p = (float **) calloc(m, sizeof(float *));
for(i=0;i<m;i++)
    p[i] = (float *) calloc(n, sizeof(float));
for(i = 0;  i<m;  i++)
   for(j = 0;  j<n; j++)
   {
       printf("p[%d][%d] = ", i+1, j+1);
       scanf("%f",&tg);
       p[i][j] = tg;
    }
printf("\nMa tran vua nhap vao:\n");
for(i = 0;  i<m; i++)
   {
      for(j = 0;  j<n; j++) printf("%8.2f",p[i][j]);
      printf("\n");
   }
for(i = 0; i<m; i++) free(p[i]);
free(p);
```

```
// Tim nhung phan tu xuat hien nhieu nhat trong day so
x1,x2,..., xn
#include <stdio.h>
#include <conio.h>
void main()
{
  float *x;
  int *dem,n,demmax,i,j,ktra;

  clrscr();
// Nhap so lieu
  printf("Nhap n : ");
  scanf("%d",&n);
  x=(float*)malloc(n*sizeof(float));
  dem=(int*)malloc(n*sizeof(int));
  for (i=0;i<n;i++)
    {
    printf("x[%d] = ",i+1);
    scanf("%f",&x[i]);
    }
// Tinh mang dem[] va tim so lan xuat hien nhieu nhat
  demmax = 0;
  for(i=0;i<n;i++)
    {
    dem[i] = 0;
    for (j=0;j<n;j++)
                if (x[j]==x[i]) dem[i]++;
    if (demmax < dem[i]) demmax = dem[i];
    }
// In ket qua
    printf("\nSo ;an xuat hien nhieu nhat la %d",demmax);
  for(i=0; i<n; i++)
   if(dem[i]==demmax)
                { ktra=1;
                  for(j=0; j<i; j++)
                                if(x[i]==x[j]) {ktra=0; break;}
                  if(ktra)printf("\nGia tri %10.2f",x[i]);
                }
  getch();
  free(x); free(dem);
}
```

```
// Ket qua bong da
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <alloc.h>
void main()
{
   int n,i,j,temp,*s,**a;
   char **ten,*tg;
   clrscr();
   do
     { printf("Nhap so doi bong da n = ");
               scanf("%d",&n);} while (n<=0);
   ten=(char **) malloc(n*sizeof(char *));
   for(i=0;i<n;i++) ten[i]=(char *) malloc(20*sizeof(char));
   printf("\nNhap ten cac doi bong:\n");
   for(i=0;i<n;i++)
     {
               printf("Ten doi[%d] = ",i+1);
               fflush(stdin);
               gets(ten[i]);
     }
   s=(int *) malloc(n*sizeof(int));
   a=(int **) malloc(n*sizeof(int *));
   for(i=0; i<n; i++) a[i]=(int *) malloc(n*sizeof(int));
   printf("\nNhap Bang diem:\n");

for(i=0;i<n-1;i++)
     {
               for(j=i+1;j<n;j++)
                         {
                         printf("Diem cua Doi %s voi Doi %s
= ",ten[i],ten[j]);
                         scanf("%d",&temp);
                         a[i][j]=temp;
                         switch (a[i][j])
                           {
                             case 6: a[j][i]=0; break;
                             case 4: a[j][i]=1; break;
                             case 3: a[j][i]=3; break;
                             case 2: a[j][i]=2; break;
                             case 1: a[j][i]=4; break;
                             case 0: a[j][i]=6; break;
                           }
                         }
               printf("\n");
     }
   for(i=0; i<n; i++)
     {  s[i]=0;
               for(j=0; j<n; j++) if(i!=j) s[i] += a[i][j];
     }
```

```
for(i=0;i<n-1;i++)
      for(j=i+1;j<n;j++)
                  if (s[i]<s[j])
                    {
                      temp=s[i];
                      s[i]=s[j];
                      s[j]=temp;
                      strcpy(tg,ten[i]);
                      strcpy(ten[i],ten[j]);
                      strcpy(ten[j],tg);
                    }

   printf("\nBANG KET QUA:\n");
   for (i=0;i<n;i++)
     {  for (j=0;j<n;j++)
                    if(i==j) printf("  X"); else printf("%3d",a[i][j]);
                  printf("\n");
     }
   printf("\nSap xep theo tong diem giam dan:\n");
   printf("Ten doi bong da     Tong diem\n");
   printf("-------------------------------\n");
   for(i=0;i<n;i++) printf("%15s %6d\n",ten[i],s[i]);
   free(s);
   for(i=0; i<n; i++) { free(a[i]); free(ten[i]);}
   free(a);
   free(ten);
   getch();
}
```

```
// Tim day so con co do dai lon nhat thoa man dieu kien
x[i]=x[i+1]
#include <stdio.h>
#include <conio.h>
#include <alloc.h>
void main()
{
  int *x,n,i,j,dmax=1,imax;
  clrscr();
  printf("Nhap so phan tu n = ");
  scanf("%d",&n);
  x=(int*)malloc(n*sizeof(int));
  for (i=0;i<n;i++)
    {
                printf("x[%d] = ",i+1);
                scanf("%d",&x[i]);
    }
    imax=i=0;
    while (i<n)
                 {
                   j = i+1;
                   while (x[i]==x[j]) j++;
                   if (dmax < j-i) {dmax = j-i; imax=i;}
                   i=j;
                 }
  printf("\nDay con can tim gom %d phan tu la :\n",dmax);
  for (i=imax; i<=imax+dmax-1; i++)
                printf("\nx[%d] = %d  ",i+1,x[i]);
  getch();
  free(x);
}
```