Com S // Cpr E // Math 5250

Numerical Analysis of High-Performance Computing

Instructor: Songting Luo

Lecture 9: More Arrays, Switch Statements, & Introduction to Pointers in C

Outline

An example: while statement

Monty Hall problem

$ISUHPC/lectures/lecture9/codes/MontyHall.c:

```c
int main()
{
   // Where is the grand prize?
   srand( time(NULL) );
   int prize = rand()%3; // 0 <= rand() <= 32767
   int notprize1 = (prize+1)%3;
   int notprize2 = (prize+2)%3;

   // Ask contestant to pick a door
   printf("\n");
   printf(" =-=-=-=-=-=-=-=-=-=-=-=-=-\n");
   printf(" ** Monty Hall Simulator **\n");
   printf(" =-=-=-=-=-=-=-=-=-=-=-=-=-\n");
   printf("\n");
```

Monty Hall problem

$ISUHPC/lectures/lecture9/codes/MontyHall.c:

```c
    int pick=-1;
    while(pick<0 || pick>2)
    {
       printf(" Pick a door (0, 1, or 2): ");
       scanf("%d", &pick);
    }

    printf("\n You entered: %d\n", pick);

    // Tell contestant about another door
    int other;
    int other_other;
    printf("\n Interesting choice ...\n");
```

Monty Hall problem

$ISUHPC/lectures/lecture9/codes/MontyHall.c:

```
    if (pick==prize)
    {
       int  ss = rand()%2;
       if (ss==0)
       {
          other = notprize1;
          other_other = notprize2;
       }
       else
       {
          other = notprize2;
          other_other = notprize1;
       }
    }
```

Monty Hall problem

$ISUHPC/lectures/lecture9/codes/MontyHall.c:

```c
    else
    {
       other_other = prize;
       if (pick==notprize1)
       {
          other = notprize2;
       }
       else
       {
          other = notprize1;
       }
    }
```

Monty Hall problem

$ISUHPC/lectures/lecture9/codes/MontyHall.c:

```c
    printf("\n I can tell you for sure that the prize is not
         behind door: %i\n",
        other);

    // Ask if they want to change
    int change = -1;
    while(change!=0 && change!=1)
    {
        printf("\n Stay with Door %i (press 0) or switch to Door
              %i (press 1): ",
            pick,other_other);
        scanf("%d", &change);
    }
```

Monty Hall problem

$ISUHPC/lectures/lecture9/codes/MontyHall.c:

```
    int final_pick;
    if (change==0)
    {
       final_pick = pick;
       printf("\n You stayed with Door %i\n ",final_pick);
    }
    else
    {
       final_pick = other_other;
       printf("\n You switched to Door %i\n ",final_pick);
    }
```

Monty Hall problem

$ISUHPC/lectures/lecture9/codes/MontyHall.c:

```c
    // Check answer
    if (final_pick==prize)
    {
       printf("\n  *** WINNER ***\n\n");
    }
    else
    {
       printf("\n  --- LOSER ---\n\n");
    }
    printf(" The prize was behind Door %i\n\n",prize);

    return 0;
 }
```

Arrays in functions

Ints/Floats/Chars are passed by value

$ISUHPC/lectures/lecture9/codes/intfunc.c:

```c
int main()
{
   int a =  2;
   int b = -5;

   printf("\n Before function call:\n");
   printf("%3i %3i \n",a,b);

   void somefunc(int a, int b);
   somefunc(a,b);

   printf("\n After function call:\n");
   printf("%3i %3i \n\n",a,b);
}

void somefunc(int a, int b)
{
   a = -7; b = 11;

   printf("\n During function call:\n");
   printf("%3i %3i \n",a,b);
}
```

Ints/Floats/Chars are passed by value

```
$ gcc intfunc.c
$ ./a.out

 Before function call:
  2  -5

 During function call:
 -7  11

 After function call:
  2  -5
```

Arrays are passed by reference

$ISUHPC/lectures/lecture9/codes/arrfunc.c:

```c
int main()
{
   int somearray[] = {-1,-2,-3,0,2,4,6};

   printf("\n Before function call:\n");
   for (int i=0; i<7; i++)
   {
      if (i>0){ printf(","); }
      printf(" %3i",somearray[i]);
   }
   printf("\n");

   void somefunc(int somearray[]);
   somefunc(somearray);

   printf("\n After function call:\n");
   for (int i=0; i<7; i++)
   {
      if (i>0){ printf(","); }
      printf(" %3i",somearray[i]);
   }
   printf("\n\n");
}
```

Arrays are passed by reference

$ISUHPC/lectures/lecture9/codes/arrfunc.c:

```c
void somefunc(int somearray[])
{
   for (int i=0; i<7; i++)
   {
      somearray[i] = -2*somearray[i];
   }
}
```

```
$ gcc arrfunc.c
$ ./a.out

 Before function call:
  -1,  -2,  -3,   0,   2,   4,   6

 After function call:
   2,   4,   6,   0,  -4,  -8, -12
```

Switch statements

Example: evaluating a basis expansion

Consider a N^{th} degree polynomial:

$$p_N(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_N x^N$$

We can write this

$$p_N(x) = \sum_{i=0}^{N} a_i \, \psi_i(x), \qquad \psi_i(x) \in \left\{1,\, x,\, x^2, \ldots,\, x^n\right\},$$

where ψ_i the i^{th} degree monomial. Therefore, we view ψ_i for $i = 0, 1, \ldots, N$ as a **basis** for polynomials of degree N.

Example: evaluating a basis expansion

Consider a N^{th} degree polynomial:

$$p_N(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_N x^N$$

We can write this

$$p_N(x) = \sum_{i=0}^{N} a_i \, \psi_i(x), \qquad \psi_i(x) \in \left\{1,\, x,\, x^2, \ldots,\, x^n\right\},$$

where ψ_i the i^{th} degree monomial. Therefore, we view ψ_i for $i = 0, 1, \ldots, N$ as a **basis** for polynomials of degree N.

In order to evaluate the polynomial at some point $x = a$, we must first evaluate each ψ_i at $x = a$ and sum from $i = 0, \ldots, N$.

Example: evaluating a basis expansion

Consider a N^{th} degree polynomial:

$$p_N(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_N x^N$$

We can write this

$$p_N(x) = \sum_{i=0}^{N} a_i\, \psi_i(x), \qquad \psi_i(x) \in \left\{1,\, x,\, x^2, \ldots,\, x^n\right\},$$

where ψ_i the i^{th} degree monomial. Therefore, we view ψ_i for $i = 0, 1, \ldots, N$ as a **basis** for polynomials of degree N.

In order to evaluate the polynomial at some point $x = a$, we must first evaluate each ψ_i at $x = a$ and sum from $i = 0, \ldots, N$.

It turns out that a better basis than monomials are the Legendre polynomials:

$$\phi_i(x) \in \left\{1,\, \sqrt{3}\,x,\, \frac{\sqrt{5}}{2}\left(3x^2 - 1\right),\, \frac{\sqrt{7}}{2}(5x^3 - 3x),\, \frac{1}{8}\left(105x^4 - 90x^2 + 9\right), \ldots\right\}$$

Then

$$p_N(x) = \sum_{i=0}^{N} b_i\, \phi_i(x).$$

Example: evaluating a basis expansion

$ISUHPC/lectures/lecture9/codes/legendre1.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main()
{
   const int Nmax = 4;
   int N;

   // read-in polynomial degree
   printf("\n Input polynomial degree (0-%i): ",Nmax);
   scanf("%i", &N);
   if (N<0 || N>Nmax)
   {
      printf(" Invalid value N = %i.\n",N);
      printf(" N must satisfy: 0 <= N <= %i\n\n",Nmax);
      exit(1);
   }
   printf("\n");
```

Example: evaluating a basis expansion

$ISUHPC/lectures/lecture9/codes/legendre1.c:

```c
    // read-in coefficients
    double b[Nmax+1];
    for (int i=0; i<=N; i++)
    {
       printf(" Set b[%i]: ",i);
       scanf("%lf", &b[i]);
    }
    printf("\n");

    // set x-coordinates
    const int NumPts = 21;
    double x[NumPts];
    for (int i=0; i<NumPts; i++)
    {
       x[i] = -1.0 + i*(2.0/(1.0*(NumPts-1)));
    }
```

Example: evaluating a basis expansion

$ISUHPC/lectures/lecture9/codes/legendre1.c:

```
    // Calculate polynomial at x-coordinates
    double y[NumPts];
    void SamplePoly(const int N,
                    const int NumPts,
                    const double b[],
                    const double x[],
                    double y[]);
    SamplePoly(N,NumPts,b,x,y);

    // Write to file
    void WritePoly(const int NumPts,
                   const double x[],
                   double y[]);
    WritePoly(NumPts,x,y);

    // Call python script to plot
    system("python3 PlotPoly.py");

    return 0;
 }
```

Example: evaluating a basis expansion

$ISUHPC/lectures/lecture9/codes/PlotPoly.py:

```
def PlotPoly():
    # First, figure out how many floats there are
    fid = open('poly.data', 'r')
    NumPts = 0;
    while True:
        line = fid.readline()
        if not line: break
        NumPts = NumPts+1
    fid.close()

    import numpy as np; import string as str;
    import matplotlib.pyplot as plt

    # Second, read-in all the floats
    x = np.zeros(NumPts,dtype=float)
    y = np.zeros(NumPts,dtype=float)
    fid = open('poly.data', 'r')
    for k in range(0,NumPts):
        linestring = fid.readline()
        #linelist   = str.split(linestring)
        linelist = linestring.split()
        x[k]       = float(linelist[0])
        y[k]       = float(linelist[1])
```

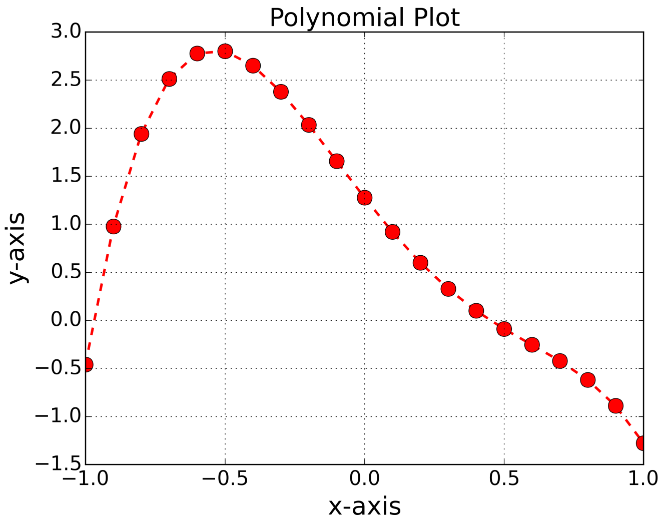
Example: evaluating a basis expansion

`$ISUHPC/lectures/lecture9/codes/PlotPoly.py`:

```python

    # Third, plot the result
    plt.rc("font", size=16); plt.figure(1)
    plt.plot(x,y,linestyle="dashed",linewidth=2,marker="o",
             color="red",markersize=12);
    plt.xlim(-1.0,1.0); plt.xticks([-1.0,-0.5,0.0,0.5,1.0])
    ax = plt.gca(); ax.grid(True)
    plt.xlabel("x-axis",size=20); plt.ylabel("y-axis",size=20)
    plt.title("Polynomial Plot",size=20)
    plt.savefig('legendre.png',dpi=400,bbox_inches='tight')
    plt.show();
if __name__ == "__main__":
    PlotPoly();
```

```
$ gcc legendre1.c; ./a.out

 Input polynomial degree (0-4): 4

 Set b[0]: 1
 Set b[1]: -1
 Set b[2]: -0.5
 Set b[3]: 0.5
 Set b[4]: -0.25
```

Example: evaluating a basis expansion



Sampling the polynomial

Selecting which basis: `$ISUHPC/lectures/lecture9/codes/legendre1.c`**:**

```
void SamplePoly(const int N,         // polynomial degree
                const int NumPts,    // number of x point values
                const double b[],    // polynomial coefficients
                const double x[],    // x point values
                double y[])          // y(x) output values
{
```

Version #1: If/else if/else if/.../else statement

Selecting which basis: $ISUHPC/lectures/lecture9/codes/legendre1.c:

```
    for (int i=0; i<NumPts; i++)
    {
       const double a = x[i];
       y[i] = b[0];
       for (int k=1; k<=N; k++)
       {
          double phi;
          if (k==1)
          { phi = sqrt(3.0)*a;                           }
          else if (k==2)
          { phi = 0.5*sqrt(5.0)*(3.0*pow(a,2)-1.0);     }
          else if (k==3)
          { phi = 0.5*sqrt(7.0)*(5.0*pow(a,3)-3.0*a);  }
          else if (k==4)
          { phi = 0.125*(105*pow(a,4)-90*pow(a,2)+9);  }
          else
          { printf("\n Error \n."); exit(1);            }

          y[i] += b[k]*phi;
       }
    }
 }
```

Version #2: Switch statement

Selecting which basis: `$ISUHPC/lectures/lecture9/codes/legendre2.c`:

```c
    for (int i=0; i<NumPts; i++)
    {
        const double a = x[i]; double phi; y[i] = b[0];
        for (int k=1; k<=N; k++)
        {
            switch(k)
            {
                case 1:
                    phi = sqrt(3.0)*a; break;
                case 2:
                    phi = 0.5*sqrt(5.0)*(3.0*pow(a,2)-1.0); break;
                case 3:
                    phi =0.5*sqrt(7.0)*(5.0*pow(a,3)-3.0*a); break;
                case 4:
                    phi =0.125*(105*pow(a,4)-90*pow(a,2)+9); break;
                default:
                    printf("\n Error \n."); exit(1);
            }
            y[i] += b[k]*phi;
        }
    }
}
```

Version #3: Cascading switch statement

Selecting which basis: `$ISUHPC/lectures/lecture9/codes/legendre3.c`:

```
    for (int i=0; i<NumPts; i++)
    {
       const double a = x[i]; double phi; y[i] = b[0];
       switch(N)
       {
          case 4:
             phi = 0.125*(105*pow(a,4)-90*pow(a,2)+9);
             y[i] += b[4]*phi;
          case 3:
             phi = 0.5*sqrt(7.0)*(5.0*pow(a,3)-3.0*a);
             y[i] += b[3]*phi;
          case 2:
             phi = 0.5*sqrt(5.0)*(3.0*pow(a,2)-1.0);
             y[i] += b[2]*phi;
          case 1:
             phi = sqrt(3.0)*a;
             y[i] += b[1]*phi;
             break;
          case 0:
             break;
          default:
             printf("\n Error \n."); exit(1);
```

Introduction to Pointers in C

Memory address

Every declared variable in C is stored somewhere in memory. You can use the & (ampersand) operator to obtain the **memory address** of that variable.

$ISUHPC/lectures/lecture9/codes/simple1.c:

```c
#include <stdio.h>

int main ()
{
   int  var1;
   char var2[10];

   printf("Address of var1 variable: %p\n", &var1 );
   printf("Address of var2 variable: %p\n", &var2 );

   return 0;
}
```

```
$ gcc simple1.c
$ ./a.out
Address of var1 variable: 0x7ffee04ffa04
Address of var2 variable: 0x7ffee04ffa0e
```

Pointers

A **pointer** is a variable whose value is the address of another variable, i.e., direct address of the memory location.

$ISUHPC/lectures/lecture9/codes/simple2.c:

```c
int main ()
{
   int  var = -11;   /* actual variable declaration */
   int  *ip;         /* pointer variable declaration */
   ip = &var;  /* store address of var in pointer variable*/

   printf("       Address of var variable: %p\n", &var );
   printf(" Address stored in ip variable: %p\n", ip   );
   printf("         Value of *ip variable: %d\n", *ip  );

   return 0;
}
```

```
$ gcc simple2.c
$ a.out
       Address of var variable: 0x7ffee4b38a18
Address stored in ip variable: 0x7ffee4b38a18
         Value of *ip variable: -11
```

NULL pointer

A **null pointer** is a special pointer value that is known not to point anywhere. What this means that no other valid pointer, to any other variable or array cell or anything else, will ever compare equal to a null pointer.

$ISUHPC/lectures/lecture9/codes/simple3.c:

```
int main()
{
   int  *ptr = NULL;
   printf("The value of ptr is : %p\n", ptr  );
   return 0;
}
```

```
$ gcc simple2.c
$ ./a.out
The value of ptr is : 0x0
```

Passing Ints/Floats/Chars by reference

$ISUHPC/lectures/lecture9/codes/intfunc_byref.c:

```c
int main()
{
   int a =  2;
   int b = -5;

   printf("\n Before function call:\n");
   printf("%3i %3i \n",a,b);

   void somefunc(int* a, int* b);
   somefunc(&a,&b);

   printf("\n After function call:\n");
   printf("%3i %3i \n\n",a,b);
}

void somefunc(int* a, int* b)
{
   *a = -7; *b = 11;

   printf("\n During function call:\n");
   printf("%3i %3i \n",*a,*b);
}
```

Passing Ints/Floats/Chars by reference

NOTE: To change the value of the variable to which the pointer is pointing to, you need to **dereference** the pointer and then treat as a regular variable:

```c
    *a = -7; *b = 11;

    printf("\n During function call:\n");
    printf("%3i %3i \n",*a,*b);
```

```
$ gcc intfunc_byref.c
$ ./a.out

 Before function call:
  2  -5

 During function call:
 -7  11

 After function call:
 -7  11
```

Lab assignment

- Monty Hall Simulation (with name MontyHall.c): submitting source code and screenshots.
- Expansion with Chebyshev polynomials:

  $$\phi_i(x) \in \{1,\ x,\ 2x^2 - 1,\ 4x^3 - 3x,\ 8x^4 - 8x^2 + 1,\ 16x^5 - 20x^3 + 5x,\ \cdots\}$$

  Then

  $$p_N(x) = \sum_{i=0}^{N} b_i\, \phi_i(x),$$

  for $0 \le N \le 5$.

  1. Using **switch** statement (with name **chebyshev.c**)
  2. Plotting the polynomial with Python, using **system()** to run Python script from C.
  3. Submit code, figure, and screenshots.
- Update Git.