Lecture 9

Using Pointers

Instructor: Ashley Gannon

ISC3313 Fall 2021

Pointers to arrays

Last class we saw how to use a pointer to a chunk of memory. This is known as *dynamic* memory.

```
int n = 10;
int* p = new int[n];
for (int i = 0; i < 10; i++)
{
p[i] = i;
}
cout<<*p;
```

Pointers to arrays

Last class we saw how to use a pointer to a chunk of memory. This is known as *dynamic* memory.

```
int n = 10;
int* p = new int[n];
for (int i = 0; i < 10; i++)
{
p[i] = i;
}
cout<<p[4];
```

Pointers to arrays

Last class we saw how to use a pointer to a chunk of memory. This is known as *dynamic* memory.

```
int n = 10;
int* p = new int[n];
for (int i = 0; i < 10; i++)
{
p[i] = i;
}
cout<<*p[4];
```

Pointers to arrays: deleting the heap

Delete can be used by either using `Delete operator` or `Delete [] operator`

Using `new` for dynamic memory allocation puts variables on heap memory. Which means Delete operator deallocates memory from heap.

NOTE: The pointer to object is not destroyed, value or memory block pointed by pointer is available to reuse.

```cpp
int n = 10;
int* p = new int[n];
for (int i = 0; i < 10; i++)
{
p[i] = i;
}
delete[] p;
```

Discretization

Discretization is the process of transferring continuous functions, models, variables, and/or equations into discrete counterparts. This process is usually carried out as a first step toward making them suitable for numerical evaluation.

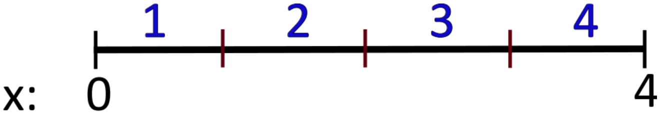
Let's consider the continuous variable x, that exists on the domain $[0, 4]$.

Discretization

Discretization is the process of transferring continuous functions, models, variables, and/or equations into discrete counterparts. This process is usually carried out as a first step toward making them suitable for numerical evaluation.

Let's consider the continuous variable x, that exists on the domain $[0, 4]$.

- We want to discretize this domain by chopping it into 4 parts.

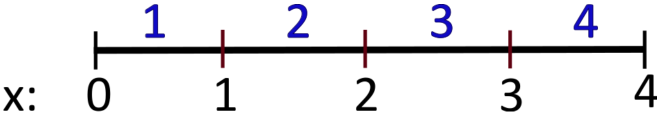


Discretization

Discretization is the process of transferring continuous functions, models, variables, and/or equations into discrete counterparts. This process is usually carried out as a first step toward making them suitable for numerical evaluation.

Let's consider the continuous variable x, that exists on the domain $[0, 4]$.

- We want to discretize this domain by chopping it into 4 parts.
- Set the values of x

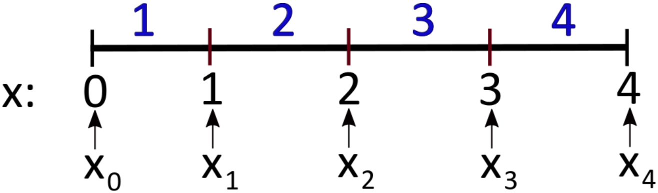


Discretization

Discretization is the process of transferring continuous functions, models, variables, and/or equations into discrete counterparts. This process is usually carried out as a first step toward making them suitable for numerical evaluation.

Let's consider the continuous variable x, that exists on the domain $[0, 4]$.

- We want to discretize this domain by chopping it into 4 elements (parts).
- Set the values of x



NOTICE: we have 4 elements, but 5 points in our array.

Class activity 1

We are going to discretize a domain $x \in [x_a, x_b]$ by chopping it into N separate elements. Each element will be of size

$$\Delta x = \frac{x_b - x_a}{N}$$

There will be $N + 1$ points. We are going to store them in an array. For integer i is 0 to N we need to compute

$$x_i = i * \Delta x$$

Let $x_a = 0$, $x_b = 20$, and $N = 20$. Compute Δx once and store it. Then use a `for` loop to compute the x_i. Print out x to verify your code is correct.

Post your code in the discussion board **Discretization Code** for participation credit.

Pointers for passing arrays

We saw how to pass a variable to a function by reference. It is possible to do this using pointers as well.

```
int n = 10;
int* p = new int[n];
myfunc(p,n);
```

where the declaration of `myfunc` might look like:

```
void myfunc(int* a, int N);
```

Class activity 2

Write a `void` function (called `linspace`) that takes as an argument

- a pointer (type `double`),
- lower and upper bounds (x_a and x_b, type `double`),
- and the number of elements (N type `int`) for the domain to be discretized by

The code should compute $N + 1$ points along the domain and store them in the array pointer. Test your code for $N = 10$, and $x_a = 0$, $x_b = 1$.

Post your code in the discussion board **Linspace Function** for participation credit.