Chapter 2

Program Elements

We seek in this section to solve mathematical problems of the type quadrature.

\[ I = \int_{a}^{b} f(x) \, dx \approx \sum_{i=0}^{N-1} f(x_i) \delta, \]

for the left hand rule. In order to compute this integral approximation, we need to decide the discretization rule for the \( x_i \)'s, for instance a uniform spacing, \( x_i = a + \delta \cdot i \), with \( \delta = \frac{b-a}{N} \). The answer is stored in a variable ‘sum’ and displayed. The pseudo-code is then given in algorithm (2).

<table>
<thead>
<tr>
<th>Algorithm 2 Left point rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Input: ( f, a, b, N )</td>
</tr>
<tr>
<td>2: ( \delta \leftarrow \frac{b-a}{N} )</td>
</tr>
<tr>
<td>3: sum = 0</td>
</tr>
<tr>
<td>4: for ( i = 0, \ldots, N - 1 ) do</td>
</tr>
<tr>
<td>5: ( x_i = a + \delta \cdot i )</td>
</tr>
<tr>
<td>6: sum = sum + ( f(x_i) \cdot \delta )</td>
</tr>
<tr>
<td>7: end for</td>
</tr>
<tr>
<td>8: Output: sum</td>
</tr>
</tbody>
</table>

2.1 Data Types

- integers
  `int i; // exact whole number`

- floating point values
  `double x; // Floating point real numbers with 15 digits`

- string Strings are not built in so you need to include a string header file.

```cpp
#include <iostream>
#include <string>
using namespace std;

int main ()
{
    string mystring = "This is a string";
    cout << mystring;
    return 0;
}
```
• boolean

```cpp
bool statement // either true or false
```

• const

```cpp
const double PI=3.141592653589793
```

### 2.2 Control Structures

#### 2.2.1 IF/THEN/ELSE

```cpp
if (logical expression)
  do stuff;
else if (logical expression)
  do other stuff;
else
  do other stuff;
```

#### 2.2.2 FOR

```cpp
#include <iostream>
using namespace std;

int main ()
{
  for (int i=1; i<3; i++) {
    cout << i;
  }
  return 0;
}
```

i++ = i=i+1  

i−− = i=i−1 

i+=a = i=i+a 

i−=a = i=i−a 

i*=a = i=i*a 

i/=a = i=i/a 

In general this is

```cpp
for (int integer=starting_value; logical_expression; increment)
{
  //CODE
}
```

#### 2.2.3 Break

```cpp
#include <iostream>
using namespace std;

int main ()
{
  for (int i=1; i<100; i+=2) {
    cout << i;
    if (i*i>20)
      i++;
  }
}
```
2.2.4 Continue

```cpp
#include <iostream>
using namespace std;
int main ()
{
    for (int i = 1; i < 5; i++) {
        if (i == 2) {
            continue;
        }
        cout << i;
    }
    return 0;
}
```

2.3 I/O

2.3.1 To Screen

- Output

```cpp
cout<<" this is output";
cout<<" this is output which ends the line"<<endl;
cout<<" this is output which ends the line\n";
```

this is output
this is output which ends the line
this is output which ends the line

- Input

```cpp
#include <iostream>
using namespace std;
int main ()
{
    int i;
    double x;
    cin>>i;
    cin>>x;
    cout<<i<<", "<<x;
    return 0;
}
```

Better form is to have output before each read to tell you what to type in. This does not warn you when you type in the wrong thing.

skipping getline for strings.
2.3.2 To File

- Output

```cpp
#include <iostream>
#include <fstream>
using namespace std;

int main () {
    double x=1.2;
    ofstream testoutput;
    testoutput.open ("testoutput.dat");
    testoutput << x<<"\n"<<x*x;
    testoutput.close();
    return 0;
}
```

- Input

```cpp
#include <iostream>
#include <fstream>
using namespace std;

int main () {
    double x,y;
    ifstream testinput;
    testinput.open ("testoutput.dat");
    testinput >> x>>y;
    cout<<x<<","<<y;
    testinput.close();
    return 0;
}
```