

1 Moderncode

Title

```
1 int main(int ac, char *av[])
2 {
3     printf("Hello, World");
4     return 0;
5 }
```

This title is very very very very
very very very very very long

```
1 int main(int ac, char *av[])
2 {
3     printf("Hello, World");
4     return 0;
5 }
```

```
1 int main(int ac, char *av[])
2 {
3     printf("Hello, World");
4     return 0;
5 }
```

This is a very long code

```
1 int main(int ac, char *av[])
2 {
3     printf("Hello, World");
4     printf("Hello, World");
5     printf("Hello, World");
6     printf("Hello, World");
7     printf("Hello, World");
8     printf("Hello, World");
9     printf("Hello, World");
10    printf("Hello, World");
11    printf("Hello, World");
12    printf("Hello, World");
13    printf("Hello, World");
14    printf("Hello, World");
15    printf("Hello, World");
16    printf("Hello, World");
```

```

17     printf("Hello, World");
18     printf("Hello, World");
19     return 0;
20 }

```

```

1  \documentclass[10pt]{article}
2  \usepackage{moderncode}
3
4  \begin{document}
5  \section{Moderncode}
6
7  \begin{moderncode}[C][adjusted title={Title}]
8  int main(int ac, char *av[])
9  {
10     printf("Hello, World");
11     return 0;
12 }
13 \end{moderncode}
14
15 \begin{moderncode}[C][adjusted title={This title
    ↪ is very very very very very very very
    ↪ very very long}]
16 int main(int ac, char *av[])
17 {
18     printf("Hello, World");
19     return 0;
20 }
21 \end{moderncode}
22
23 \begin{moderncode}[C]
24 int main(int ac, char *av[])
25 {
26     printf("Hello, World");
27     return 0;
28 }
29 \end{moderncode}
30
31 \begin{moderncode}[C][adjusted title={This is a
    ↪ very long code}]
32 int main(int ac, char *av[])
33 {
34     printf("Hello, World");
35     printf("Hello, World");

```

```

36     printf("Hello, World");
37     printf("Hello, World");
38     printf("Hello, World");
39     printf("Hello, World");
40     printf("Hello, World");
41     printf("Hello, World");
42     printf("Hello, World");
43     printf("Hello, World");
44     printf("Hello, World");
45     printf("Hello, World");
46     printf("Hello, World");
47     printf("Hello, World");
48     printf("Hello, World");
49     printf("Hello, World");
50     return 0;
51 }
52 \end{moderncode}
53
54 \moderncodeinput[TeX][]{example.tex}
55
56 \subsection{Output}
57
58 \begin{moderncodeout}
59 Enter a positive integer: 100
60 Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21,
    ↪ 34, 55, 89
61 \end{moderncodeout}
62
63 \subsection{Inline}
64
65 \subsubsection{Inline Code}
66
67 This is an inline modern code display: \
    ↪ moderncodeinline{\LaTeX}. It is also
    ↪ possible to define a language: \
    ↪ moderncodeinline[SQL]{SELECT pg_relation_
    ↪ size('title_basics');}.
68
69 \subsection{Inline Key}
70
71 It also supports a key-like-style inline element
    ↪ : \moderncodekey{Ctrl} + \moderncodekey{C}
72
73 \section{Lstlisting}

```

```

74
75 \begin{lstlisting}[caption=Example in C++,
    ↪ language=c++]
76 #include <iostream>
77 using namespace std;
78
79 int main() {
80     int n, t1 = 0, t2 = 1, nextTerm = 0;
81
82     cout << "Enter the number of terms: ";
83     cin >> n;
84
85     cout << "Fibonacci Series: ";
86
87     for (int i = 1; i <= n; ++i) {
88         // prints the first two terms.
89         if(i == 1) {
90             cout << t1 << ", ";
91             continue;
92         }
93         if(i == 2) {
94             cout << t2 << ", ";
95             continue;
96         }
97         nextTerm = t1 + t2;
98         t1 = t2;
99         t2 = nextTerm;
100
101         cout << nextTerm << ", ";
102     }
103     return 0;
104 }
105 \end{lstlisting}
106
107 \subsection{Output}
108
109 \begin{lstlisting}[style=lstoutput]
110 Enter a positive integer: 100
111 Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21,
    ↪ 34, 55, 89
112 \end{lstlisting}
113
114 \subsection{Inline}
115

```

```

116 \subsubsection{Inline Code}
117
118 \lstinline{\LaTeX}
119
120 \end{document}

```

1.1 Output

```

Enter a positive integer: 100
Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,
↪ 55, 89

```

1.2 Inline

1.2.1 Inline Code

This is an inline modern code display: `\LaTeX`. It is also possible to define a language: `SELECT pg_relation_size('title_basics');`.

1.3 Inline Key

It also supports a key-like-style inline element: `Ctrl + C`

2 Lstlisting

Listing 1: Example in C++

```

1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int n, t1 = 0, t2 = 1, nextTerm = 0;
6
7      cout << "Enter the number of terms: ";
8      cin >> n;
9
10     cout << "Fibonacci Series: ";
11
12     for (int i = 1; i <= n; ++i) {
13         // prints the first two terms.
14         if(i == 1) {
15             cout << t1 << ", ";
16             continue;
17         }

```

```

18         if(i == 2) {
19             cout << t2 << ", ";
20             continue;
21         }
22         nextTerm = t1 + t2;
23         t1 = t2;
24         t2 = nextTerm;
25
26         cout << nextTerm << ", ";
27     }
28     return 0;
29 }

```

2.1 Output

Output

```

Enter a positive integer: 100
Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,
↵ 55, 89

```

2.2 Inline

2.2.1 Inline Code

\LaTeX