1.3 THE *char* TYPE

The *char* type is used for working **with characters**. A date of this type will represent a single character. To store more characters we will use an **array** with *char* elements or a *string*.

Variables of type char

A char variable is declared like this:

e.g.: *char C;*

The value of a *char* (or *signed char*) variable is a natural number between **-128** and **127**. Values between **0** and **127** correspond to characters in the **ASCII code**.

Similarly, *unsigned char* data has values between **0** and **255**. We notice that both types contain the values that correspond to the characters in the **ASCII code**.

Literals

A *char literal* (value) is a **character** in the **ASCII code**, delimited by **apostrophe** characters (').

We can initialize a *char* variable by assigning it a **char literal** or a **numeric value**. If the numeric value does not belong to the corresponding value range, it will be **truncated**.

e.g.: *char C;*
C = 'A';
C = 65;

Display and reading

Although *char* data stores integers, reading and displaying them will work **with characters**.

Display

main.cpp

```cpp
#include <iostream>
using namespace std;
int main ()
{
    char C = 'A';
    cout << C; // A
    C = 65;
    cout << C; // A
}
```

Displaying a char data is done like this:

- if the value is **from ASCII code**, the corresponding character will be displayed. For non-printable characters the effect depends on the character and the working environment (maybe nothing or various symbols are displayed).
- if the value is **outside the ASCII code**, the effect depends on the working environment.

Reading

main.cpp

```cpp
#include <iostream>
using namespace std;
int main ()
{
    char C;
    cin >> C;
    cout<<C;
}
```

After reading a *char* variable from the keyboard, it will represent the character entered. If more than one character is entered, **only the first character will be read**.

Examples

1. main.cpp

```cpp
#include <iostream>
using namespace std;
int main ()
{
    char x;
    cin >> x; // enter A
    cout << x; // A
}
```

2. main.cpp

```cpp
#include <iostream>
using namespace std;
int main ()
{
    char x;
    cin >> x; // enter 145
    cout << x; // 1
}
```

3. main.cpp

```cpp
#include <iostream>
using namespace std;
int main ()
{
    char x, y;
    cin >> x >> y; // enter A B
    cout << x << endl; // A
    cout << y << endl; // B
}
```

4.

main.cpp

```cpp
#include <iostream>
using namespace std;
int main ()
{
    char x, y;
    cin >> x >> y; // enter AB
    cout << x << endl; // A
    cout << y << endl; // B
}
```

5.

main.cpp

```cpp
#include <iostream>
using namespace std;
int main ()
{
    char x, y;
    cin >> x >> y; // enter ABC
    cout << x << endl; // A
    cout << y << endl; // B
}
```

6.

main.cpp

```cpp
#include <iostream>
using namespace std;
int main ()
{
    char x, y;
    cin >> x >> y; // enter 65 66
    cout << x << endl; // 6
    cout << y << endl; // 5
}
```

Operations. Type conversions

char values can be converted to **other** types.

e.g.:

main.cpp

```cpp
#include <iostream>
using namespace std;
int main ()
{
    char x;
    x = 65; // default conversion from int to char
    cout << x; // A
    cout << (int) x; // 65
    int n = 65;
    cout << (char) n; //A
}
```

All the usual number operations can be done with *char* data. The *char* value will be converted to *int* by default, **then** the operations will be done.

e.g.:

1.

main.cpp

```cpp
#include <iostream>
using namespace std;
int main ()
{
    char x = 'A';
    cout << x + 1; // 66
    cout << (char)(x + 1); // B
}
```

2.

main.cpp

```cpp
#include <iostream>
using namespace std;
int main ()
{
    char x = 'A';
    x ++;
    cout << x; // B
}
```

Convert between upper and lower case

A frequent problem is determining, for an uppercase letter, the corresponding lowercase letter, or vice versa. The solution is based on the fact that in **the ASCII code**, uppercase letters are positioned **before** lowercase ones, and the difference between the ASCII code of a lowercase letter and the ASCII code of the corresponding uppercase letter is the same for all letters (32).

The transformation will be done by **subtracting** this value from the lowercase letter, or by adding it to the uppercase letter.

e.g.:

main.cpp

```cpp
#include <iostream>
using namespace std;
int main ()
{
    int dif = 'a' - 'A'; // 32
    char x = 'k';
    x = x - dif;
    cout << x; // K
}
```

Pay attention to!

- " – quotes delimit strings. A single-character string is not the same as a character. ("A" ≠ 'A'!)