GAGAN KHATRI
itsgagankhatri@gmail.com

JANUARY 7, 2023

10 Best Practices for Coding in C

GREENIE WEB

Codes GitHub Repo
https://github.com/ItsGagan/10BestPracticesforC

Contents :

1 BASIC BEST PRACTICES, LIKE ...

These Following Four are the Most Important, Disscussion of other Uncommon Best Practices without depicting these won't be fair enough

A. Using Comments to give Better Understanding of Code.
B. Using Functions to Segregate tasks & Organize Code Properly.
C. Choosing Proper Datatypes (like here float for Percentage is required).
D. Giving meaningful Names to Variables and Functions.

```
float getPercentage(int obtainedMarks, int totalMarks) {
    // Calculate the percentage by dividing the obtained marks by the total number of
marks, and multiplying by 100
    float percentage = (obtainedMarks / (float)totalMarks) * 100;
    // Return the percentage
    return percentage;
}
```

FULL CODE LINK : Click Here

2 USING POINTERS CAREFULLY ...

A. Initialization while declaration : It should either be initialized to some required valid location or NULL.

B. Memory Management : The memory referenced by the pointer should be freed after its desired use.

```
    // Declare an integer variable and a pointer to an integer, initialized with the
address of the integer variable
    int x = 10;
    int *ptr = &x;

    // Modify the value of the integer through the pointer
    *ptr = 20;

    // Free the memory pointed to by the pointer
    free(ptr);
```

FULL CODE LINK : Click Here

3 STATIC FUNCTION - CONST FUNCTION ...

A. In C, a **static** function is a function whose scope is limited to the file in which it is defined. This means that the function can only be called from within the file in which it is defined, and cannot be called from other files.
A **static** function can be useful for creating functions that are only needed within a particular file, and do not need to be accessed from other parts of the program.

B. In C, a **const** function is a function that does not modify the object on which it is called. This means that the function cannot change any of the member variables of the object, or any other variables in the program.
A **const** function can be useful for creating functions that are guaranteed to not modify any data in the program, which can be helpful for ensuring the integrity of the data.

FULL CODE LINK : Static Function and Const Function

4 INLINE FUNCTION ...

In C, an **inline** function is a function that is expanded in line at the point where it is called, rather than being called as a separate function. This can be useful for improving the performance of a program, as it reduces the overhead of calling a function.

```
inline int add(int x, int y) {
    return x + y;
}
```

- The compiler is free to ignore the inline keyword and treat the function as a normal function, depending on the optimization level and other factors.

- The compiler is free to ignore the inline keyword and treat the function as a normal function, depending on the optimization level and other factors.

FULL CODE LINK : Click Here

5 USING 'RESTRICT' KEYWORD ...

In C, the **restrict** keyword is used to indicate that a pointer is the only way to access a particular piece of memory. This can be useful for optimizing code, as the compiler can assume that the memory pointed to by the **restrict** pointer is not accessed through any other means.

```c
void swap(int *restrict a, int *restrict b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

The **restrict** keyword can be useful for optimizing code when it is known that a particular piece of memory is only accessed through a single pointer.

FULL CODE LINK : Click Here

6 USING GOTO (SPARINGLY) ...

The **goto statement** in C allows the program to jump to a different point in the code, specified by a label.
The goto statement is also considered to be a poor programming practice, as it can make code more difficult to understand and maintain. Hence use it wisely...

```c
#include <stdio.h>

int main() {
    int i = 0;
    loop:
    printf("Iteration %d\n", i);
    i++;
    if (i < 10) {
        goto loop;
    }
    return 0;
}
```

In this example, the **goto** statement is used to jump back to the beginning of the loop, causing the loop to repeat until the **i** variable is greater than or equal to 10.

In some cases, the goto statement can be useful for breaking out of deeply nested loops or for implementing a finite state machine.

FULL CODE LINK : Click Here

7

USING 'CONST' KEYWORD ...

In C, the const keyword is used to indicate that a variable or object is constant, meaning that its value cannot be modified. This can be useful for ensuring the integrity of data, as it prevents accidental or intentional modifications to the data.

```c
#include <stdio.h>


int main() {
    const int x = 10;
    x = 20;
    return 0;
}
```

In C, the const keyword is used to indicate that a variable or object is constant, meaning that its value cannot be modified. This can be useful for ensuring the integrity of data, as it prevents accidental or intentional modifications to the data.

```c
#include <stdio.h>


int main() {
    const int x = 10;
    const int *ptr = &x;
    *ptr = 20;
    return 0;
}
```

Both Codes will Result into Compile Error, as Const won't let it modify...

FULL CODE LINK : Code 1 , Code 2

8 DANGLING PROBLEM IN POINTERS ...

Dangling Pointer Problem : while assigning one pointer to another (shallow copy). One may end up 'free'ing one copy while the other one still points to that location. If copying is the objective of a task, go for deep copy to avoid this problem.

```c
#include <stdio.h>


int main() {
    int *ptr;
    ptr = (int*)0x1000;
    printf("*ptr = %d\n", *ptr);
    return 0;
}
```

This will result in a segmentation fault, as the pointer is pointing to an invalid memory address

FULL CODE LINK : Click Here

9 USING HEADER GUARDS ...

Header guards are a preprocessor construct used to prevent the contents of a header file from being included more than once in a single translation unit. This is important, as including the same header file multiple times can cause redefinition errors, as the same symbols may be defined multiple times.

```
#ifndef MY_HEADER_H
#define MY_HEADER_H

void foo();

#endif
```

```
#include "my_header.h"

int main() {
    foo();
    return 0;
}
```

The #ifndef directive checks if the symbol HEADER_FILE_NAME_H is not defined, and if it is not, defines it. The #endif directive ends the header guard.

When the source file is compiled, the preprocessor will include the contents of my_header.h, but only the first time it is encountered. Any subsequent #include directives for the same header file will be ignored.

10 USING 'ASSERT' ...

In C, the **assert** macro is used to perform runtime checks for debugging purposes. The **assert** macro takes an expression as an argument, and if the expression is false, prints an error message and aborts the program.

```c
#include <assert.h>


int main() {
    int x = 10;
    assert(x > 0);
    assert(x < 20);
    return 0;
}
```

In this example, the first assert call will succeed, as the value of x is indeed greater than 0. The second assert call will also succeed, as the value of x is less than 20.

The assert macro is typically used to perform basic sanity checks during development, to ensure that the program is in a valid state. It is not intended to be used as a replacement for proper error handling in a production program.

FULL CODE LINK : Click Here

BONUS POINTS

- Follow coding standards: Adhere to established coding standards and style guides to ensure that your code is consistent and easy to read.

- Test your code thoroughly: Make sure to test your code thoroughly to ensure that it is correct and that it works as intended.

- Use version control: Use a version control system such as Git to track changes to your code and collaborate with others. This can help to make it easier to manage your code and collaborate with others.

THANK YOU SO MUCH