5.6 — Do while statements

BY ALEX ON JUNE 25TH, 2007 | LAST MODIFIED BY ALEX ON JANUARY 23RD, 2020

One interesting thing about the while loop is that if the loop condition is initially false, the while loop will not execute at all. It is sometimes the case that we know we want a loop to execute at least once, such as when displaying a menu. To help facilitate this, C++ offers the do-while loop:

```
do
    statement
while (condition);
```

The statement in a do-while loop always executes at least once. After the statement has been executed, the do-while loop checks the condition. If the condition is true, the path of execution jumps back to the top of the do-while loop and executes it again.

Here is an example of using a do-while loop to display a menu to the user and wait for the user to make a valid choice:

```cpp
#include <iostream>

int main()
{
    // selection must be declared outside do/while loop
    int selection;

    do
    {
        std::cout << "Please make a selection: \n";
        std::cout << "1) Addition\n";
        std::cout << "2) Subtraction\n";
        std::cout << "3) Multiplication\n";
        std::cout << "4) Division\n";
        std::cin >> selection;
    }
    while (selection != 1 && selection != 2 &&
        selection != 3 && selection != 4);

    // do something with selection here
    // such as a switch statement

    std::cout << "You selected option #" << selection << "\n";

    return 0;
}
```

One interesting thing about the above example is that the selection variable must be declared outside of the do block. Why do you think that is?

If the selection variable were to be declared inside the do block, it would be destroyed when the do block terminates, which happens before the while conditional is executed. But we need the variable to use in the while conditional -- consequently, the selection variable must be declared outside the do block.

Generally it is good form to use a do-while loop instead of a while loop when you intentionally want the loop to execute at least once, as it makes this assumption explicit -- however, it's not that big of a deal either way.

5.7 -- For statements

Index

5.5 -- While statements

C++ TUTORIAL | PRINT THIS POST

89 comments to 5.6 — Do while statements

« Older Comments 1 2

james
September 28, 2019 at 9:35 am · Reply

```
#include <stdio.h>


int main (void)
{
    char menu_option;


    printf("        PROGRAM TO MEASURE AN AREA\n");
    printf("------------------------------------------\n\n");


    do{
    printf("\n\nMain Menu\n");
    printf("a. Area of Parallelogram\n");
    printf("b. Area of a Square\n");
    printf("c. Area of Triangle\n");
    printf("d. Area of Trapezoid\n");
    printf("e. Area of Circle\n");
    printf("f. Exit\n\n");
    printf(" Please enter an option from the main menu: ");
    scanf("%c",&menu_option);

    switch(menu_option){

    case 'a':
        {
      float base,height,area;
      printf("enter base of parallelogram: ");
      scanf("%f",&base);

      printf("enter height of the parallelogram: ");
      scanf("%f",&height);

      area=(base*height);
      printf("Area Of Parallelogram: %f\n",area);
      //return 0;

         }

        break;

    case 'b':
        {
        int square_side, area;

        printf("Enter the side of square: ");
        scanf("%d", &square_side);

        //calculation of the area
        area = square_side * square_side;

        printf("Area of the square: %d", area);
        //return 0;

        }

        break;

    case'c':
        {
        float base, height, area;

        printf("\n Please Enter the Base of a Triangle  :  ");
        scanf("%f", &base);

        printf("\n Please Enter the Height of a Triangle  :  ");
        scanf("%f", &height);

        area = (base * height) / 2;

        printf("\n The Area of a Triangle using Base and Height = %.2f\n", area);

        //return 0;

            }
        break;
    case'd':

        {
        int height;
        int b1,b2; //breadth1 and breath2
        long area_of_trapezoid;

        // clrscr();

        printf(" Program to calculate Area of a Trapezoid : \n\n ");
        printf(" Enter the Height of a Trapezoid : ");
        scanf("%d", &height);
        printf("\n\n Enter the Breadth1 value : ");
        scanf("%d", &b1);
        printf("\n\n Enter the Breadth2 value : ");
        scanf("%d", &b2);

        area_of_trapezoid = (height/2)*(b1+b2);

        printf("\n\n Area of Trapezoid with Height = %d, Breadth1 = %d and Breadth2 = %d =
        // getch();

        //return 0;


        }
        break;

    case'e':
        {
        int circle_radius;
        float PI_VALUE=3.14, circle_area, circle_circumf;

         //Ask user to enter the radius of circle
        printf("\nEnter radius of circle: ");
        //Storing the user input into variable circle_radius
        scanf("%d",&circle_radius);

        //Calculate and display Area
        circle_area = PI_VALUE * circle_radius * circle_radius;
        printf("\nArea of circle is: %f",circle_area);

        //Calculate and display Circumference
        circle_circumf = 2 * PI_VALUE * circle_radius;
        printf("\nCircumference of circle is: %f",circle_circumf);

        //return(0);


        }

        break;

    case'f':
        //break;
     {
    //default:
        printf("Exiting Code : Bye ");
        //return(0);
           // break;
     }
     //break;
    }
    //return 0;
    } while(menu_option !='f');
}
```

can someone tell me why the menu is repeating twice after the answer is showing?, it should show the menu again after the answer is calculated?

nascardriver
September 28, 2019 at 11:50 pm · Reply

When you enter

```
x<enter>
```

Where <enter> is you pressing the enter key, the input stream stores

```
x\n
```

Where '\n' in a line feed. When you call `scanf("%c", &ch)`, you're asking for one character to be extracted. `ch` is set to 'x', the input stream still holds

```
\n
```

The next time you call `scanf("%c", &menu_option)`, that \n is still there and gets stored in `menu_option`. If you want the first call to remove the '\n', you can add it to the format string.

```
scanf("%c\n", &ch);
```

I don't know C so I can't tell if this has any side effects, but it seems to work from my tests.

Bombi Barlsson
October 5, 2019 at 10:02 pm · Reply

I'm not very proficient at C++, but from what I can tell

```
std::cin.clear();
std::cin.ignore(32767,'\n');
```

works as a solution. I believe ignore() was mentioned in a previous tutorial. Clear removes the error flag for cin in case there's an invalid input type, while ignore ignores any character in the next 32767 characters until the next line. It's a type of flushing mechanism, and I tend to use it for my inputs for this tutorial but I have no idea if it's proper form.

Luiz
November 3, 2019 at 11:20 am · Reply

```
cin.ignore(numeric_limits<streamsize>::max(), '\n')
```

will ignore everthing until it hits EOF or a new line

Benur21
August 2, 2019 at 7:04 am · Reply

Wouldn't a normal while work as good as this one?
Like this:

```
#include <iostream>
```

```
2
3   int main()
4   {
5       // selection must be declared outside do/while loop
6       int selection;
7
8       while (selection != 1 && selection != 2 &&
9           selection != 3 && selection != 4) {
10            std::cout << "Please make a selection: \n";
11            std::cout << "1) Addition\n";
12            std::cout << "2) Subtraction\n";
13            std::cout << "3) Multiplication\n";
14            std::cout << "4) Division\n";
15            std::cout >> selection;
16          }
17
18      // do something with selection here
19      // such as a switch statement
20
21      std::cout << "You selected option #" << selection << "\n";
22
23      return 0;
24  }
```

Another question: Why when I input a letter for the selection, like "r", it will loop forever?

nascardriver
August 2, 2019 at 8:54 am · Reply

You're accessing an uninitialized variable in line 8 and 9, this causes undefined behavior. You want the first cycle of the loop to be unconditional, so use a do-while-loop.

> Why when I input a letter for the selection, like "r", it will loop forever?
'r' cannot be extracted into an `int`. Extraction fails, the stream enters a failed state and all further operations on it are ignored. This is covered later in this chapter.

Parsa
July 29, 2019 at 3:15 pm · Reply

Why isn't this working?
(I enter -1 as input but it doesn't go back to the do loop.

```
 int main()

{
    i16 chooseNumber{};


    do
    {
        std::cout << "choose an integer: ";
        std::cin >> chooseNumber;
    }
    while (chooseNumber >= 0.0);

    return 0;
}
```

nascardriver
July 29, 2019 at 11:27 pm · Reply

The loop only runs while `chooseNumber >= 0.0`. -1 isn't greater than 0.

Parsa
July 31, 2019 at 7:26 am · Reply

Oh right, thanks.

Anastasia
July 21, 2019 at 6:47 am · Reply

Hi! First I would like to thank Alex and nascardriver for the wonderful work you are doing, it's the best structured and comprehensive C++ course I've ever seen. Now, I'm a C++ beginner and I wrote a little counting coins programm summarizing the latest concepts I've learned. Could somebody review it for me, please? I have some doubts about whether I've handled certain aspects correctly (or at least at an acceptable level) in particular the exit of the selection choices, the default case of the switch (what would be the best way to end the execution there?) and whether it's acceptable to create a new struct every time the while statement executes (in the printSum() function). Thank you!

```cpp
#include <iostream>
#include <string>

enum class Coin_type {
    PENNY,
    NICKEL,
    DIME,
    QUARTER,
    HALF,
    DOLLAR
};
struct Coin {
    Coin_type type;
    std::string name;
    int value;
};
Coin getCoin() {
    int selected_coin{0};

    do {
        std::cout << "Please, choose your coin or enter 0 to exit:\n";
        std::cout << "0) exit\n";
        std::cout << "1) Penny\n";
        std::cout << "2) Nickel\n";
        std::cout << "3) Dime\n";
        std::cout << "4) Quater\n";
        std::cout << "5) Half dollar\n";
        std::cout << "6) Dollar\n";
        std::cin >> selected_coin;
    } while(selected_coin < 0 || selected_coin > 6);

    if(selected_coin != 0)
        std::cout << "Your coin is ";
    else {
        Coin coin = {};
        return coin;
    }

    switch(selected_coin) {
        case 1:
            {
                Coin penny = {Coin_type::PENNY, "penny", 1};
                std::cout << penny.name << "\n";
                return penny;
            }
        case 2:
            {
                Coin nickel = {Coin_type::NICKEL, "nickel", 5};
                std::cout << nickel.name << "\n";
                return nickel;
            }
        case 3:
            {
                Coin dime = {Coin_type::DIME, "dime", 10};
                std::cout << dime.name << "\n";
                return dime;
            }
        case 4:
            {
                Coin quarter = {Coin_type::QUARTER, "quarter", 25};
                std::cout << quarter.name << "\n";
                return quarter;
            }
        case 5:
            {
                Coin half = {Coin_type::HALF, "half dollar", 50};
                std::cout << half.name << "\n";
                return half;
            }
        case 6:
            {
                Coin dollar = {Coin_type::DOLLAR, "dollar", 100};
                std::cout << dollar.name << "\n";
                return dollar;
            }
        default:
            std::cout << "unidentified\n";
            //is there a better way to handle this?
            exit(1);
    }
}
void printSum() {
    int sum {0};
    bool exit {false};
    do {
        //creating a new struct with each iteration
        Coin coin = getCoin();

        //the exit is not explicit enough?
        if(coin.value == 0)
            exit = true;
        else
            sum += coin.value;
    } while(!exit);

    std::cout << "You have $" << static_cast<double>(sum)/100 << "\n";
}
int main() {

```

```
100         printSum();
101
102         return 0;
103     }
```

nascardriver
July 21, 2019 at 7:07 am · Reply

Hi!

- Initialize your variables with brace initializers.
- Line 23-30 and 40-70 should use the indexes of `Coin_type`.
- Line 35, 36: `return {};`.
- Line 43, 49, 55... are all the same, move them out of the switch.
- Create a coin before the switch-statement, assign to it inside the switch, return after the switch.
- Use single quotation marks for characters.
- Line 93 causes undefined behavior, `sum` is uninitialized.
- Use double literals for doubles (100.0 instead of 100).
- Inconsistent formatting, use your editor's auto-formatting feature.

> is there a better way to handle this?
Yes, you'll learn about `assert` later.

> creating a new struct with each iteration
You're not creating a "new struct", you're creating a new instance of a struct.

> the exit is not explicit enough?
It's fine. You could also use a `while (true)` loop and `break` to remove the need of `exit`.

Anastasia
July 21, 2019 at 10:48 am · Reply

Thank you so much for your response, nascardriver, it helps a lot, I hope I understood everything right.

> Initialize your variables with brace initializers.
> Line 93 causes undefined behavior, `sum` is uninitialized.
I did initialize it to 0 with brace initialization (line 85 above). Did I do it wrong?

> Line 23-30 and 40-70 should use the indexes of `Coin_type`.
With an enum class the compiler doesn't allow to convert an enumerator to int and it would require to use static_cast to use the indexes. I changed Coin_type to a simple enum and it works this way.

Hopefully this looks a bit better now:

```
#include <iostream>
#include <string>

enum Coin_type { PENNY = 1, NICKEL, DIME, QUARTER, HALF, DOLLAR };

struct Coin {
  Coin_type type;
  std::string name;
  int value;
};
Coin getCoin() {
  int selected_coin{0};

  do {
    std::cout << "Please, choose your coin or enter 0 to exit:\n";
    std::cout << "0) exit" << '\n';
    std::cout << "1) Penny" << '\n';
    std::cout << "2) Nickel" << '\n';
    std::cout << "3) Dime" << '\n';
    std::cout << "4) Quater" << '\n';
    std::cout << "5) Half dollar" << '\n';
    std::cout << "6) Dollar" << '\n';
    std::cin >> selected_coin;
  } while (selected_coin < 0 || selected_coin > 6);

  if (selected_coin != 0)
    std::cout << "Your coin is ";
  else {
    return {};
  }
  Coin coin = {};

  switch (selected_coin) {
  case Coin_type::PENNY:
    coin = {Coin_type::PENNY, "penny", 1};
    break;
  case Coin_type::NICKEL:
    coin = {Coin_type::NICKEL, "nickel", 5};
    break;
  case Coin_type::DIME:
    coin = {Coin_type::DIME, "dime", 10};
    break;
  case Coin_type::QUARTER:
    coin = {Coin_type::QUARTER, "quarter", 25};
    break;
  case Coin_type::HALF:
    coin = {Coin_type::HALF, "half dollar", 50};
    break;
  case Coin_type::DOLLAR:
    coin = {Coin_type::DOLLAR, "dollar", 100};
    break;
  default:
    std::cout << "unidentified" << '\n';
    exit(1);
  }
  std::cout << coin.name << '\n';
  return coin;
}
void printSum() {
  int sum{0};
  bool exit{false};
  do {
    Coin coin = getCoin();

    if (coin.value == 0)
      exit = true;
    else
      sum += coin.value;
  } while (!exit);

  std::cout << "You have $" << sum / 100.0 << '\n';
}
int main() {

  printSum();

```

```
77      return 0;
78  }
```

nascardriver
July 22, 2019 at 5:33 am · Reply

> I did initialize it to 0 with brace initialization (line 85 above). Did I do it wrong?
No, that's correct. Either I missed it or you edited it after/while I looked at your code.

- Line 31, 63: Initialize your variables with brace initializers.

Line 17-22 should use the enumerators.
Line 24 should use the enumerators.
Line 35, 38, ...: You already know the type of the coin, there's no need to repeat it

```
coin = { static_cast<Coin_type>(selected_coin), "dollar", 100 };
```

Anastasia
July 22, 2019 at 6:37 am · Reply

> Either I missed it or you edit it after/while I looked at your code.
No, you didn't miss it. I did in fact edit this line shortly after posting. I was just worrying that it is not the right way(or place) to initialize it.

> Line 17-22 should use the enumerators.
I wasn't sure whether I should use enumerators here, because the code is just asking for the user's input.

I think I fixed everything this time. Won't post it, because I'm afraid I've already taken too much space.

Thank you one more time for your help and patience.

Envy
February 28, 2019 at 2:40 am · Reply

I just skimmed the comments and there were only several questions if we couldn't replace the && with || in do-while loop, which is obviously wrong. But what I thought about is why won't we use

```
while (selection < 1 || selection > 4);
```

instead of

```
while (selection != 1 && selection != 2 && selection != 3 && selection != 4);
```

If we would add a 5th option to choose from you had to add another "&& selection != 5", whereas in my example you just had to change the 4 into a 5. I know this only works here because we use consecutive numbers but it just bothered me a bit :D

Walter
July 10, 2019 at 2:44 am · Reply

You're right, this would work, because we use numbers as stand-in for explicit options. Selection 2 is 'Subtraction', but it's also the number 2. It makes sense to say "2 + 1 is 3", but you can't say "Subtraction + Addition is Multiplication".

We should be using an enum for the options here, but for brevity, this has been excluded. The following doesn't really make sense, for example:

```
while (selection < Selection::addition || selection > Selection::division)
```

I believe this is the reason the conditional is written the way it is.

SamiraFerdi
October 29, 2018 at 8:49 pm · Reply

Alex and nascardriver, can you check this for me?

```
/*PROGRAM TO DISPLAY ALL EVEN OR ODD NUMBERS IN A CERTAIN RANGE
 *
 *This program is run by two step:
 *  1. Checking if the user entering a random even or odd number and display the result
 *  2. User entering any the maximum range and display all even/odd number in that range
 *     based on the result of point 1 above
 * PS. Sorry for my bad english and bad comment
 * PS. If I entering number that higher than 1 in runAgain variable, my program run foreve
 */

#include <iostream>

//From step 1
//Check for the number of user entered even or odd number
bool isEvenOrOdd(int number);
//Display the result: even or odd number
void displayEvenOrOddResult (int number, bool result);

//From step 2
//Display All Even or Odd Number in a certain range that user input
void displayAllEvenOrOddNumber(int range, bool result);

int main()
{
  //Do you want to run the program again? 1 = run again / 0 = quit the program
  bool runAgain;

  do
  {
    //Please check my newline formatting. Is there a better way?
    std::cout << "===PROGRAM TO DISPLAY ALL EVEN OR ODD NUMBERS IN A CERTAIN RANGE===" <<

    std::cout << "Enter a even or odd number: ";
    int number;
    std::cin >> number;

    std::cout << "\n";

    bool result = isEvenOrOdd(number);
    displayEvenOrOddNumberResult(number, result);

    std::cout << "\n";

    std::cout << "Enter a certain range: ";
    int range;
    std::cin >> range;

    std::cout << "\n";

    displayAllEvenOrOddNumber(range, result);

    std::cout << "\n";

    std::cout << "Do you wanna run the program again? (1 = run again / 0 = quit the progra
    std::cin >> runAgain;

    std::cout << "\n";

  } while(runAgain);

  return 0;
}

//The functions definiton

bool isEvenOrOdd(int number)
{
  bool result{number % 2 == 0};

  switch (result)
  {
    case true:
      return true;
      break;

    //Or using case keyword
    default:
      return false;
  }
}

void displayEvenOrOddNumberResult (int number, bool result)
{

//Just another way to output the result. Should I use this instead of switch-case?
//  if (result)
//    std::cout << number << " is a even number" << std::endl;
//  else
//    std::cout << number << " is a odd number" << std::endl;

  switch (result)
  {
    case true:
      std::cout << number << " is a even number" << std::endl;
      break;

    //Or maybe use default keyword for false condition
    case false:
      std::cout << number << " is a odd number" << std::endl;
      break;
  }
}

void displayAllEvenOrOddNumber(int range, bool result)
{
  switch (result)
  {
    case true:
      std::cout << "All even number in range of " << range << " is ";
      for (int evenNumber = 0; evenNumber <= range; evenNumber+=2)
        std::cout << evenNumber << " ";
```

```
   break;

  //Or maybe use default keyword for false condition
  case false:
   std::cout << "All odd number in range of " << range << " is ";
   for (int oddNumber = 1; oddNumber <= range; oddNumber+=2)
    std::cout << oddNumber << " ";
   break;
 }
}
```

nascardriver
October 31, 2018 at 7:36 am · Reply

* Initialize your variables with uniform initialization
* Unnecessary forward declarations. Move @main below the other functions
* @isEvenOrOdd can be done in 1 line. Give it another try
* You don't need to switch a bool. Use an if-statement
* @displayEvenOrOddNumberResult can also be done in 1 line

SamiraFerdi
October 31, 2018 at 5:04 pm · Reply

Thank you for reply!

what about this?

```cpp
#include <iostream>
#include <string>

//Return true if even number, and false if odd number
bool isEven(int number)
{
 return (number % 2 == 0);
}

std::string displayResult(bool number)
{
 return { (number) ? "even number\n" : "odd number\n" };
}

int main()
{
 std::cout << "===PROGRAM TO TEST A NUMBER IS EVEN OR ODD NUMBER===" << "\n\n";
 std::cout << "Enter a number: ";
 int number{ 0 };
 std::cin >> number;

 std::cout << "\n\n";

 //Test for number
 bool testNumber{ isEven(number) };
 std::cout << number << " is " << displayResult(testNumber);

 return 0;
}
```

nascardriver
November 1, 2018 at 5:38 am · Reply

That's a lot better!
You don't need @displayResult, you can do it in-line

```
std::cout << number << " is an " << (testNumber ? "even" : "odd") << " number\
```

SamiraFerdi
November 1, 2018 at 8:20 pm · Reply

Thank you nascardriver!

I'm in love with alex's tutor
August 20, 2018 at 3:36 am · Reply

Good day! Dear Mr Alex/NAS
 Can we change nested while statements to a do..while ? I kindly & respectfully ask you to show me how to change nested while loops that you previously showed up in section 5.5 as shown below.I have been trying my best but i cant execute what i need.

```
#include <iostream>

// Loop between 1 and 5
int main()
{
    int outer = 1;
    while (outer <= 5)
    {
        // loop between 1 and outer
        int inner = 1;
        while (inner <= outer)
            std::cout << inner++ << " ";

        // print a newline at the end of each row
        std::cout << "\n";
        ++outer;
    }

    return 0;
}
```

God Bless you all!!!!

Alex
August 22, 2018 at 12:12 pm · Reply

```
#include <iostream>

// Loop between 1 and 5
int main()
{
    int outer = 1;
    do
    {
        // loop between 1 and outer
        int inner = 1;
        do
            std::cout << inner++ << " ";
        while (inner <= outer);

        // print a newline at the end of each row
        std::cout << "\n";
        ++outer;
    } while (outer <= 5);

    return 0;
}
```

I'm in love with alex's tutor
August 22, 2018 at 11:13 pm · Reply

My hero Mr Alex!
I very++ thank you to open my eyes to see the path of programming world. I am addicted to your tutorial really.

```
for(long long count=1;count>0;++count)
{
    std::cout<< "thank you Mr. Alex";
std::cout<<"\n";
}
//infinite thank you
```

Nigel Booth
August 2, 2018 at 5:59 am · Reply

Hi Alex,

Borrowed your menu and made a simple calculator with it:

```
// Calculator_2.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <iostream>

void add()
{
    double x;
    double y;

    std::cout << "please enter a number: ";
    std::cin >> x;
    std::cout << "please enter a second number: ";
    std::cin >> y;
    std::cout << "\n" << "The result of " << x << " + " << y << " is " << x + y << "\n";
}

void subtract()
{
    double x;
    double y;

    std::cout << "please enter a number: ";
    std::cin >> x;
    std::cout << "please enter a second number: ";
    std::cin >> y;
    std::cout << "\n" << "The result of " << x << " - " << y << " is " << x - y << "\n";
}

void times()
{
    double x;
    double y;

    std::cout << "please enter a number: ";
    std::cin >> x;
    std::cout << "please enter a second number: ";
    std::cin >> y;
    std::cout << "\n" << "The result of " << x << " * " << y << " is " << x * y << "\n";
}

void divide()
{
    double x;
    double y;

    std::cout << "please enter a number: ";
    std::cin >> x;
    std::cout << "please enter a second number: ";
    std::cin >> y;
    std::cout << "\n" << "The result of " << x << " / " << y << " is " << x / y << "\n";
}

int main()
{
    // selection must be declared outside do/while loop
    int selection;

    do
    {
        std::cout << "Please make a selection: \n";
        std::cout << "1) Addition\n";
        std::cout << "2) Subtraction\n";
        std::cout << "3) Multiplication\n";
        std::cout << "4) Division\n";
        std::cin >> selection;
    } while (selection != 1 && selection != 2 &&
        selection != 3 && selection != 4);
    switch (selection)
    {
    case 1:
            add();
            break;
    case 2:
            subtract();
            break;
    case 3:
            times();
            break;
    case 4:
            divide();
            break;
    default:
        break;
    }
    return 0;
}
```

Star Light
May 1, 2018 at 11:23 pm · Reply

Your website is a treasure that I believe it's must-known for every C++ learner. I always recommend it for anyone how wanna begin with C++. Really love it.

Btw, I have an idea. It is accepting a number arguments/parameters when ran program. This one is easy. But I want to make the program auto-restart when it take a wrong parameter. Anyway, let check the code:

```
int main(int argc, char* argv[])
{
    bool running = false;
    do // I know this loop won't do the trick
    {// But I just add it for demonstration.
        if (argc == 5)//Check parameters
        {// appName -flag value -flag(parameter) value(argument)
            //Bla bla bla
        }
        else if (argc == 6)
        {// appName -flag value -flag value -flagOnly(argument)
            //Bla bla bla
        }
        else
        {
            help();// Already built outside the code
            running = true;// This one won't work, in fact, it will cause the loop bug if u
        }
    }
    while (running)
}
```

I've read the previous lesson about GOTO statement. But, in the end, those args only fetch when program starts. However, I don't know the return code to make program auto-restart. Or do we have other function for this case? And I would like to avoid: (replace "cls" with my name software)

```
system("cls");
```

Because people said antivirus don't like it and I have to register path to window var environment. And if user change software name then it die :p

I hope you could gimme some advice for this.

nascardriver
May 2, 2018 at 3:13 am · Reply

Hi Star Light!

There are ways to restart you program, but there's no point in doing so programmatically, because you need the user to pass new arguments.
If you want your program to keep running even if the user entered an erroneous command line you need to manually ask the user for input (@std::cin, @std::getline) and parse it.
This will require the user to re-enter the entire command line whereas they'd only need to press the up arrow and correct their mistake if you let them restart the application.

Star Light
May 4, 2018 at 10:35 pm · Reply

Yeah, you have a point. But I really want input as an array. std::cin or std::getline could do that, but it isn't as good as taking arguments right from the start.

« Older Comments 1 2