Week 1: Basic C++

Author: Nguyen Anh Khoa Instructor: Phuong

Basic coding in C

> *Create a simple application using the following operators:* `+, -, *, /, <<, >>`

I came up with a calculator application, input the math and it will return the result, and print out.

```c
#include <stdio.h>
#include <stdlib.h>

/*
  Lập trình C++ với các yêu cầu sau:
  - Chương trình có 4 chương trình con: cộng, trừ, nhân, chia
  - Sử dụng 4 chương trình con cộng, trừ, nhân, chia và hai lệnh dịch bit (left sh
ift, right shift) trong hàm main
*/

int plus(int x, int y) {
  return x + y;
}


int minus(int x, int y) {
  return x - y;
}


int mul(int x, int y) {
  return x * y;
}


int division(int x, int y) {
  if (y == 0) {
    return 0;
  }
  else {
    return x / y;
  }
}
```

```c
int shiftLeft(int x, int y) {
  return x << y;
}


int shiftRight(int x, int y) {
  return x >> y;
}


int main(int argc, char** argv) {
  int x = 0;
  int y = 0;
  char op = 0;
  int result = 0;

  int s = 0;
  while (s != 3) {
    printf("NUM OP NUM\n");
#if defined(_WIN32) || defined(WIN32)
    s = scanf_s("%d %c %d", &x, &op, 1, &y);
#else
    s = scanf("%d %c %d", &x, &op, &y);
#endif
  }

  if (op == '+') {
    result = plus(x, y);
  }
  else if (op == '-') {
    result = minus(x, y);
  }
  else if (op == '*') {
    result = mul(x, y);
  }
  else if (op == '/') {
    result = division(x, y);
  }
  else if (op == '<') {
    result = shiftLeft(x, y);
  }
  else if (op == '>') {
    result = shiftRight(x, y);
  }

  printf("%d %c %d = %d\n", x, op, y, result);
  return 0;
}
```

Create a program do as follows:

1. *Create static arrays of char, short, and int*
2. *Randomly generate values for three arrays*
3. *Sort the arrays*

It was hard to do it in pure C, because we would need to write 3 function to handle different types arrays for every operation. It is possible though, using C `#define` structure, but it would be hard to successfully implement one. In C++, `template` is the way. Declare the function as:

```
template<class T>
void foo(T* array);
```

To sort the arrays, I chose merge sort, it is easy to implement and efficient.

```
#include <cstdio>
#include <cstdlib>
#include <ctime>
#include <climits>
#include <iostream>


/*
Lập trình chương trình với các yêu cầu sau:
- Khai báo 3 mảng tĩnh: Mảng char, short và int
- Tạo ngẫu nhiên các ký phần tử trong 3 mảng đó
- Sắp xếp 3 mảng
*/


template<class T>
void gen_random(T* arr, size_t len);

template<class T>
void sort(T* arr, size_t len);

template<class T>
void print_array(T* arr, size_t len);

using std::cout;
using std::cin;
using std::endl;

int main(int argc, char** argv) {
  srand(time(0));

  size_t size;
  cout << "Enter the size of array: ";
  cin >> size;

  char char_arr[size];
  short short_arr[size];
  int int_arr[size];
```

```cpp
  gen_random(char_arr, size);
  gen_random(short_arr, size);
  gen_random(int_arr, size);

  cout << "After generation" << endl;
  print_array(char_arr, size);
  print_array(short_arr, size);
  print_array(int_arr, size);

  sort(char_arr, size);
  sort(short_arr, size);
  sort(int_arr, size);

  cout << "After sort" << endl;
  print_array(char_arr, size);
  print_array(short_arr, size);
  print_array(int_arr, size);
  return 0;
}

int randrange(int lower, int upper) {
  return (rand() % (upper - lower + 1)) + lower;
}

template<class T>
void gen_random(T* arr, size_t len) {
  for (size_t i = 0; i < len; i++) {
    arr[i] = (T)randrange(0, 100);
  }
}

template<class T>
void swap(T* x, T* y) {
  T temp = *x;
  *x = *y;
  *y = temp;
}

template<class T>
void sort(T* arr, size_t len) {
  if (len <= 1)
    return;
  if (len == 2) {
    if (arr[0] > arr[1])
      swap(&arr[0], &arr[1]);
    return;
  }

  T* left = arr;
  T* right = arr + len/2;
  size_t left_bound = len/2;
```

```
    size_t right_bound = len-len/2;

    sort(left, left_bound);
    sort(right, right_bound);

    // merge
    size_t left_cursor = 0;
    size_t right_cursor = 0;
    T* tmp = new T[len];
    size_t idx = 0;

    while (left_cursor < left_bound && right_cursor < right_bound) {
      if (left[left_cursor] > right[right_cursor]) {
        tmp[idx++] = right[right_cursor];
        right_cursor++;
      }
      else {
        tmp[idx++] = left[left_cursor];
        left_cursor++;
      }
    }

    for (;left_cursor < left_bound; left_cursor++)
      tmp[idx++] = left[left_cursor];
    for (;right_cursor < right_bound; right_cursor++)
      tmp[idx++] = right[right_cursor];

    for (idx = 0; idx < len; idx++)
      arr[idx] = tmp[idx];
  }

  template<class T>
  void print_array(T* arr, size_t len) {
    cout << "[";
    for (size_t i = 0; i < len; i++) {
      cout << " " << (int)arr[i] << " ";
    }
    cout << "]" << endl;
  }
```

C++ with struct

> *Create a simple library application which uses static array to store the books. The application should allow user to add or delete books.*

As the library is a static array, there is no new and delete, we need someway to know whether the book is avaiable or not, hence the structure of the book is:

```
#define author_len 21 // one for string terminator
#define title_len 51 // one for string terminator
```

```c
struct Book {
  char author[author_len];
  char title[title_len];
  char inUse;
};

typedef struct Book Book;
```

We're aming first for C, C does not allow const in array initialization, so we #define the length of the book title, and the author name. We also have two variables, `num_book` and `capacity` to check current books in library and whether or not the operation, add or delete, is possible.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


/*
Viết chương trình quản lý sách, trong đó sách có các thuộc tính sau:
1. Tên sách
2. Tác giả
Lưu ý: Không dùng mảng động

Giao diện console có các lựa chọn sau:
- Thêm sách
- Xoá sách
*/


// const at array declare is not permitted in C
#define author_len 21 // one for string terminator
#define title_len 51 // one for string terminator
#define capacity 10

struct Book {
  char author[author_len];
  char title[title_len];
  char inUse;
};

typedef struct Book Book;

Book library[capacity];
int num_book = 0;

enum CHOICES {ADDBOOK = 1, DELETEBOOK, VIEWBOOK};

void addBook();
void deleteBook();

int main(int argc, char** argv) {
    int choice = 0;
    int s = 0;
    while (1) {
      printf("LIBRARY\n");
      printf("1. add a book\n");
      printf("2. delete a book\n");
      printf(">> ");
      s = scanf("%d", &choice);
      if (s != 1) {
        printf("Please input again!");
        getchar();
        continue;
      }
      switch (choice) {
        case ADDBOOK:
          addBook();
          break;
        case DELETEBOOK:
          deleteBook();
          break;
        default:
          break;
      }
    }
    return 0;
  }

  void addBook() {
    if (num_book >= capacity) { // watch for overflow
      // Either
      // 1. Increase capacity, create new array to store books
      // 2. Fail
      printf("The library is full, no more book can be added, delete some book\n");
      getchar();
      return;
    }
    char title[title_len];
    char author[author_len];
    printf("Enter book title: ");
    scanf("%50s", title);
    printf("Enter author's name: ");
    scanf("%20s", author);

    printf("Add book %s by %s", title, author);
    char ok;
    while ((ok = getchar()) == '\n' || ok == '\r' || ok == EOF);
    if (ok != 'y') {
      printf("The operation has been cancled\n");
      printf("Press enter to continue");
      getchar();
      return;
    }
  int idx;
  for (idx = 0; idx < capacity; idx++)
    if (library[idx].inUse == 0)
      break;

  Book* newBook = &library[idx];
  strcpy(newBook->author, author);
  strcpy(newBook->title, title);
  newBook->inUse = 1;

  printf("The book is added with id: %d\n", idx);
  num_book++;
}


void xoaSach() {
  if (num_book <= 0)
    return;
  int idx;
  printf("Enter book Id that needs deletion: ");
  scanf("%d", &idx);

  if (idx < 0 || idx >= capacity) {
    printf("Id is not appropriate\n");
    return;
  }

  Book* deleteBook = &library[idx];
  if (deleteBook->inUse == 0) {
    printf("The book is not present\n");
    return;
  }

  printf("Delete book %s by %s? (y/n)", deleteBook->title, deleteBook->author);
  char ok = 0;
  while ((ok = getchar()) == '\n' || ok == '\r' || ok == EOF);
  if (ok != 'y') {
    printf("The operation has been cancled\n");
    printf("Press enter to continue");
    getchar();
    return;
  }

  deleteBook->inUse = 0;
  strcpy(deleteBook->author, "");
  strcpy(deleteBook->title, "");
  printf("Successfully delete book with Id: %d\n", idx);
  num_book--;
}
```

C++ OOP

Write classes to represent 3 different types of Bee - Worker, Queen and Drone. Each Bee has a floating-point health property, which is not writable externally and upon creation is set to a value of 100 (percent). Each Bee has a Damage() method that takes a single integer parameter that should be a value between 0 and 100. When this method is called, the health of the bee is to be reduced by that percentage of their current health. When a Worker has a health below 70% it cannot fly and therefore is pronounced Dead. When a Queen has a health below 20%, or a Drone below 50%, it is pronounced dead. This 'Dead' property should be readable from each Bee. When a Bee is dead, no further health deductions should be recorded by the bee, although the Damage() method should still be invokable without error. Your application should create a single list containing 10 instances of each type of Bee and store in a list or array. It must support methods to allow Damage() to be called for each Bee, and to return the health status of each bee, including whether it is alive or not. Your application interface must contains 2 functions. User press 1 or 2 to activate this function:

1. *Create bee list - Clear current bee list and create new random bees, then display in the console windows*
2. *Attack bees - Attack current bee list , a different random value between 0 and 80 should be selected for each bee and applied with a call to Damage(). After attacked, the user interface should refresh to show the health status of the bees in console windows Other requirements: Application type: Console We first build the basic class for Bee.*

```
enum bee_t {WORKER, QUEEN, DRONE};
class Bee {
  public:
    Bee() {}


    void Damage(int percentage) {
      if (percentage < 0 || percentage > 80)
        return;
      if (!this->_isDead)
        this->health -= this->health * ((float)percentage / 100);
      if (this->health < this->deadThreshold)
        _isDead = true;
    }


    const float& fetchHealth() const {
      return this->health;
    }
    const bool& isDead() const {
      return this->_isDead;
    }
    const char* type() const {
      switch (this->_type) {
          case WORKER:
            return "worker";
          case QUEEN:
            return "queen";
          case DRONE:
            return "drone";
          default:
            return "bee";
        }
      }
      float isAlive() {
        return !this->_isDead;
      }


    protected:
      bool _isDead;
      float health = 100;
      float deadThreshold;
      enum bee_t _type;
  };
```

Bee is an abstract class, we have three class for each type of Bee: Worker, Queen, Drone.

```
  class Worker : public Bee {
    public:
    Worker() {
      deadThreshold = 70;
      _type = WORKER;
    }
  };


  class Queen : public Bee {
    public:
    Queen() {
      deadThreshold = 20;
      _type = QUEEN;
    }
  };


  class Drone : public Bee {
    public:
    Drone() {
      deadThreshold = 50;
      _type = DRONE;
    }
  };
```

The rest of the program should be like this:

```
enum choice {CREATE = 1, ATTACK, EXIT};

const int bee_count = 30;
Bee** bee_list;

void print_bee_list();
void create_bee_list();
void attack_bee_list();

int main(int argc, char** argv) {
  srand(time(0));
  bee_list = new Bee*[bee_count];


  int choice = 0;
  int s;
  while (1) {
    print_bee_list();
    printf("1. Create bee list\n");
    printf("2. Attack bee\n");
    printf("3. Exit\n");
    s = scanf("%d", &choice);
    if (s != 1) {
      printf("Input again\n");
      getchar();
      continue;
    }


    switch(choice) {
      case CREATE:
        create_bee_list();
        break;
      case ATTACK:
        attack_bee_list();
        break;
      case EXIT:
        for (int i = 0; i < bee_count; i++) {
          delete bee_list[i];
        }
        delete[] bee_list;
        return 0;
      default:
        break;
    }
  }
  return 0;
}


void print_bee_list() {
  printf("[");
```

```
    for (int i = 0; i < bee_count; i++) {
      Bee* bee = bee_list[i];
      if (!bee)
        continue;
      printf(" %s:%d:%f ", bee->type(), bee->isDead(), bee->fetchHealth());
    }
    printf("]\n");
  }


  void create_bee_list() {
    for (int i = 0; i < bee_count; i++) {
      delete bee_list[i];
      // bee_t beetype = (bee_t)(rand() % 3);
      bee_t beetype = (bee_t)(i / 10);
      Bee* bee;
      switch (beetype) {
        case WORKER:
          bee = new Worker();
          break;
        case QUEEN:
          bee = new Queen();
          break;
        case DRONE:
          bee = new Drone();
          break;
        default:
          break;
      }
      bee_list[i] = bee;
    }
  }


  void attack_bee_list() {
    for (int i = 0; i < bee_count; i++) {
      int percent = rand() % 81;
      bee_list[i]->Damage(percent);
    }
  }
```

C++ with inline ASM

Write the function in C++ Basic operator using ASM

Visual Studio compiler and GCC compiler offer a way to write ASM code directly in our code, those are called inline ASM.

In Visual Studio, MSVC, we can write ASM in many ways:

```
// write in block
```

```
__asm {
}
// one instruction each
__asm inst op1, op2
// list instruction
__asm inst op1, op2 __asm inst op1, op2
```

The formal rule is:

```
asm-block:
    __asm assembly-instruction ;opt
    __asm { assembly-instruction-list } ;opt
assembly-instruction-list:
    assembly-instruction ;opt
    assembly-instruction ; assembly-instruction-list ;opt
```

We can use variables, labels of our code in `__asm`. For example:

```
void foo(x) {
    // move x to register eax
    __asm mov eax, x
}
```

The list of things in C you can use in inline ASM, MSVC

- Symbols, including labels and variable and function names
- Constants, including symbolic constants and enum members
- Macros and preprocessor directives
- Comments (both `/* */` and //)
- Type names (wherever a MASM type would be legal)
- typedef names, generally used with operators such as PTR and TYPE or to specify structure or union members

The series of Inline ASM for MSVC is here: Microsoft/CPP/assembler/inline (https://docs.microsoft.com/en-us/cpp/assembler/inline/inline-assembler?view=vs-2019)

Re-write the functions using `__asm` we have

```
int plus(int x, int y) {
    int retval;
    __asm {
        mov eax, x
        add eax, y
        mov retval, eax
    }
    return retval;
}

int minus(int x, int y) {
    int retval;
    __asm {
        mov eax, x
        sub eax, y
        mov retval, eax
    }
    return retval;
}

int mul(int x, int y) {
    int retval;
    __asm {
        mov eax, x
        mul y
        mov retval, eax
    }
    // overflow to edx, but sizeof(int) == sizeof(eax)
    return retval;
}

int division(int x, int y) {
    int retval;
    __asm {
        mov eax, y
        test eax, eax
        je label1        ; div zero
        mov edx, 0
        mov eax, x
        mov ecx, y
        div ecx
        mov retval, eax
    label1:
    }
    return retval;
}

/* from intel manual:
 * shift value could be imm(8) or CL (ecx low)
 */
int shiftLeft(int x, int y) {
    int retval;
    __asm {
        mov eax, x
        mov ecx, y
        shl eax, cl
        mov retval, eax
    }
    return retval;
}

int shiftRight(int x, int y) {
    int retval;
```

```
        __asm {
            mov eax, x
            mov ecx, y
            shr eax, cl
            mov retval, eax
        }
        return retval;
    }
```

The full source code could be found online: github/nganhkhoa/viettel_internship (https://github.com/nganhkhoa/viettel_internship) .

A few things different from GCC and MSVC I found when trying to compile in Windows using MSVC.

It seems that MSVC doesn't do zero initialization, boolean start with value 1, array of pointer `T**` returns a `[0xcdcdcdcd, ...]`. So when doing variable initialization:

```
bool isFalse = false;
bool isTrue = true;
bool isFalseOrTrue;     // unknown

int** array_pointer = new int*[100]; // unknown value
int** array_pointer = new int*[100] {}; // zero init, NULL
```

GCC is also too generous, we can do array size on non-const variable. While MSVC yields an error.

```
int size;
cin >> size;

int arr[size]; // GCC good, MSVC bad
```