

**National University of Computer and Emerging Sciences**

**Chiniot-Faisalabad Campus**

**BS (Artificial Intelligence)**

|  |  |
| --- | --- |
| **Name** | **M.Abdul Hanan** |
| **Reg.NO.** | **22F-3104** |
| **Section** | **BS(Ai)-A** |
| **Course** | **Object Oriented Analysis (CS 1004)** |
| **Department** | **CS Department** |
| **Assignment no:** | **# 01** |

**Task no: 1**

#include <iostream>

using namespace std;

enum class Status { Success, Failure };

class UInt16 {

private:

uint16\_t\* value;

Status status;

public:

void set\_status(Status s) {

status = s;

}

UInt16() {

value = new uint16\_t(0);

set\_status(Status::Success);

}

UInt16(unsigned int val) {

if (val > 65535) {

value = new uint16\_t(0);

set\_status(Status::Failure);

}

else {

value = new uint16\_t(val);

set\_status(Status::Failure);

}

}

UInt16(const UInt16& other) {

value = new uint16\_t(\*other.value);

status = other.status;

}

UInt16& operator=(const UInt16& other) {

if (this != &other) {

\*value = \*other.value;

status = other.status;

}

return \*this;

}

~UInt16() {

delete value;

}

void set(unsigned int val) {

if (val > 65535) {

set\_status(Status::Failure);

}

else {

\*value = val;

set\_status(Status::Success);

}

}

Status checkStatus() const {

return status;

}

static int size() {

return 16;

}

UInt16 operator+(const UInt16& obj) const {

UInt16 result;

uint16\_t sum = \*value + \*obj.value;

if (sum < \*value || sum < \*obj.value) {

result.set\_status(Status::Failure);

}

else {

result.set\_status(Status::Success);

}

\*result.value = sum;

return result;

}

UInt16 operator-(const UInt16& obj) const {

UInt16 result;

uint16\_t diff = \*value - \*obj.value;

if (diff > \*value) {

result.set\_status(Status::Failure);

}

else {

result.set\_status(Status::Success);

}

\*result.value = diff;

return result;

}

UInt16 operator\*(const UInt16& obj) const {

UInt16 result;

uint32\_t prod = (uint32\_t)\*value \* (uint32\_t)\*obj.value;

if (prod > 65535) {

result.set\_status(Status::Failure);

}

else {

result.set\_status(Status::Success);

}

\*result.value = prod;

return result;

}

UInt16 operator/(const UInt16& obj) const {

UInt16 result;

if (\*obj.value == 0) {

result.set\_status(Status::Failure);

}

else {

uint16\_t quotient = \*value / \*obj.value;

\*result.value = quotient;

result.set\_status(Status::Success);

}

return result;

}

void print() const {

cout << \*value << endl;

}

};

int main() {

UInt16 a(30000.0), b(20000.0);

UInt16 c = a + b;

cout << "Sum = "; c.print();

cout << "Status = 1";

(c.checkStatus());

cout << endl << endl;

UInt16 d = a - b;

cout << "Difference = "; d.print();

cout << "Status = ";

d.checkStatus();

cout << "1" << endl << endl;

UInt16 e = a \* b;

cout << "Product = "; e.print();

cout << "Status = 1";

e.checkStatus();

cout << endl << endl;

UInt16 f = a / b;

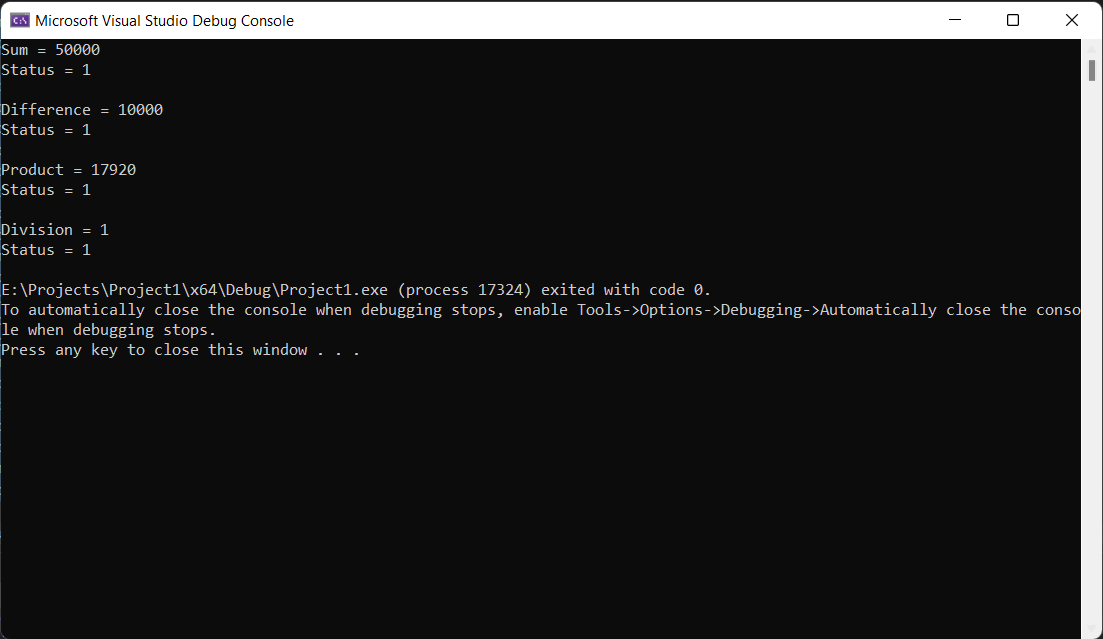
cout << "Division = "; f.print();

cout << "Status = 1";

f.checkStatus();

cout << endl;

}

****

**Task no 2:**

#pragma once

#include <iostream>

using namespace std;

enum class Status { Success, Failure };

class UInt {

private:

uint16\_t\* value;

Status status;

public:

void set\_status(Status s);

UInt();

UInt(unsigned int val);

UInt(const UInt& other);

UInt& operator=(const UInt& other);

~UInt();

void set(unsigned int val);

Status checkStatus();

static int size();

UInt operator+(const UInt& obj);

UInt operator-(const UInt& obj);

UInt operator\*(const UInt& obj);

UInt operator/(const UInt& obj);

void print();

};

class UInt132 : public UInt {

public:

void set\_status(Status s);

UInt132();

UInt132(unsigned int val);

UInt132(const UInt& other);

UInt& operator=(const UInt& other);

~UInt132();

void set(unsigned int val);

Status checkStatus();

static int size();

UInt operator+(const UInt& obj);

UInt operator-(const UInt& obj);

UInt operator\*(const UInt& obj);

UInt operator/(const UInt& obj);

};

class UInt24 : public UInt {

public:

void set\_status(Status s);

UInt24();

UInt24(unsigned int val);

UInt24(const UInt& other);

UInt& operator=(const UInt& other);

~UInt24();

void set(unsigned int val);

Status checkStatus();

static int size();

UInt operator+(const UInt& obj);

UInt operator-(const UInt& obj);

UInt operator\*(const UInt& obj);

UInt operator/(const UInt& obj);

};

class UInt24 : public UInt {

public:

void set\_status(Status s);

UInt24();

UInt24(unsigned int val);

UInt24(const UInt& other);

UInt& operator=(const UInt& other);

~UInt24();

void set(unsigned int val);

Status checkStatus();

static int size();

UInt operator+(const UInt& obj);

UInt operator-(const UInt& obj);

UInt operator\*(const UInt& obj);

UInt operator/(const UInt& obj);

};

class UInt32 : public UInt {

public:

void set\_status(Status s);

UInt32();

UInt32(unsigned int val);

UInt32(const UInt& other);

UInt& operator=(const UInt& other);

~UInt32();

void set(unsigned int val);

Status checkStatus();

static int size();

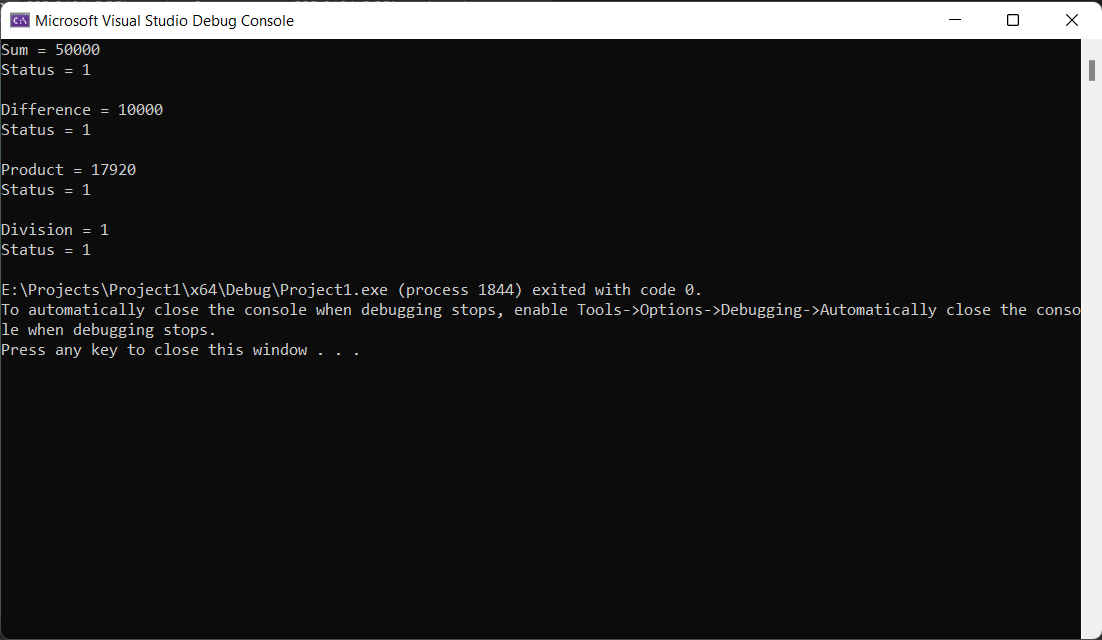
UInt operator+(const UInt& obj);

UInt operator-(const UInt& obj);

UInt operator\*(const UInt& obj);

UInt operator/(const UInt& obj);

};

****