**Subject: PRF192- PFC**

**Workshop 02**

**Objectives:**

Practicing skills at analyzing and implementing simple programs

***Contents: 7 programs***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Program** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| **Mark** | **2** | **2** | **1** | **1** | **2** | **1** | **1** |

**Program 1 ( 2 marks)**

Write a program that allows user inputting a simple expression containing one of four operators +, -, \*, / then the result is printed out to the monitor. Input format: num1 operator num2,

An example of user interface

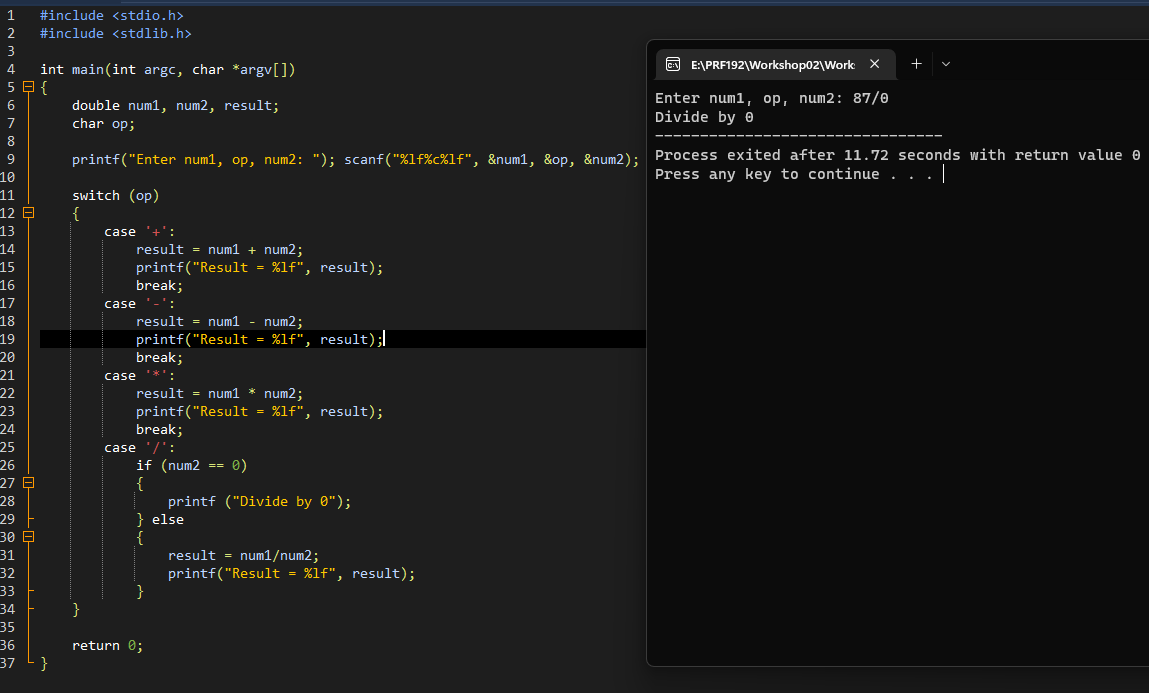
Enter an expression (+ - \* /): 4\*5

Result: 20

**Sample Analysis**

|  |  |  |
| --- | --- | --- |
|  | Content | Implementation |
| ***Nouns*** | Expression,  format num1 operator num2  result | double num1, num2  char op  double result |
| ***Verbs*** | Begin  Accept num1, op, num2  **Calculate result**  Print out result  End | scanf( “%lf%c%lf”, &num1, &op, &num2)  switch (op)  { case ‘+’ : result = num1 + num2;  print out result;  break;  case ‘-’ : result = num1 - num2;  print out result;  break;  case ‘\*’ : result = num1 \* num2;  print out result;  break;  case ‘/’ : if ( num2==0)  print out “Divide by 0 “  else  { result = num1 / num2;  print out result;  }  break;  default: print out “Op is not supported”  } |

Implement this program.



**Program 2 ( 2 marks) – Yearly Personal Income Tax**

Suppose that:

In Viet Nam, each people has to pay for his/her yearly personal income tax as the following description:

**Rules:**

***Tax-free income:***

Personal pending amount (tiền nuôi bản thân) **pa=** 9 000 000$/month

Alimony (tiền cấp dưỡng) for each his/her dependent **pd=** 3 600 000$/month/dependent

With **n** dependents, Yearly tax-free income: **tf = 12\*(pa + n\*pd)**

***Taxable income (thu nhập chịu thuế)***

**ti = income – tf**

**( If ti<=0 then income tax = 0)**

**Based on taxable income, the employee has to pay his/her income tax with levels pre-defined in the following table:**

|  |  |  |
| --- | --- | --- |
| ***Level*** | ***Taxable Income*** | ***Income tax*** |
| 1 | Less than or equal to 5.000.000 | 5% |
| 2 | From 5.000.001 to 10.000.000 | 10% |
| 3 | From 10.000.001 to 18.000.000 | 15% |
| 4 | Over 18.000.000 | 20% |

Write a program which will compute income tax of a people using the following interface:

***Case 1:***

Your income of this year: 240000000

Number of dependent:4

Tax-free income: 280800000

Taxable income: 0

Income tax: 0

***Case 1:***

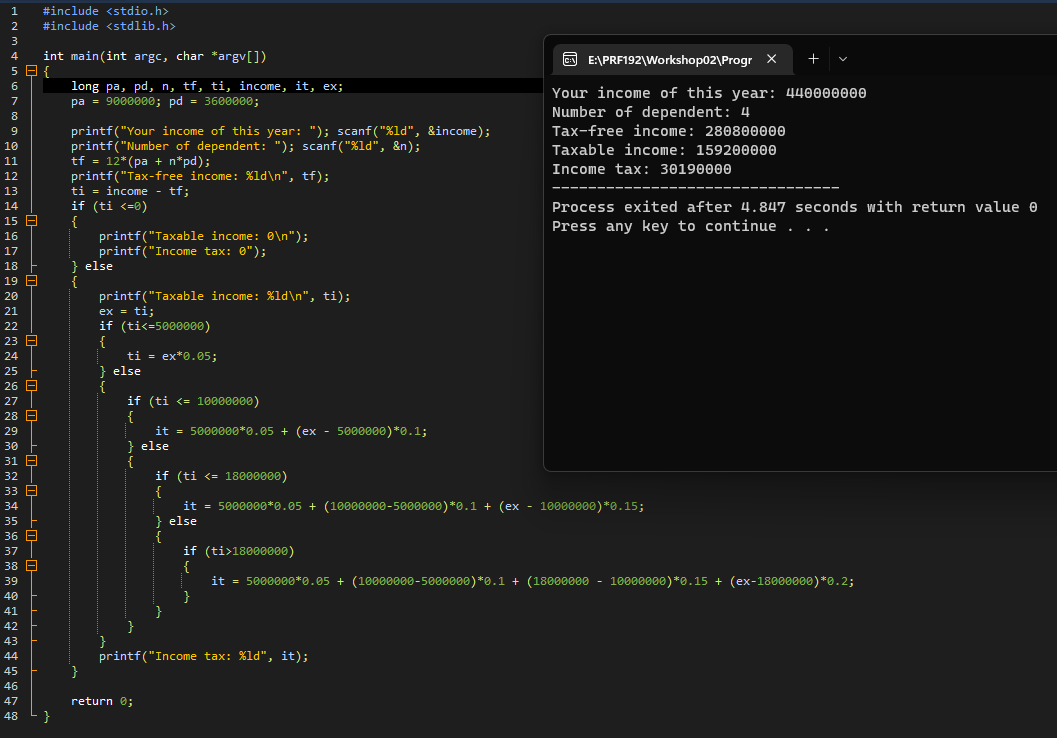
Your income of this year: 440000000

Number of dependent:4

Tax-free income: 280800000

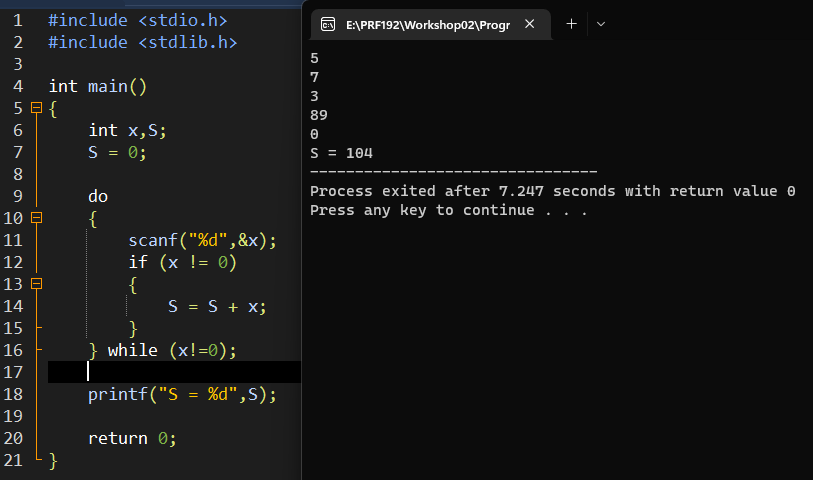
Taxable income:: 159200000

Income tax: 30190000



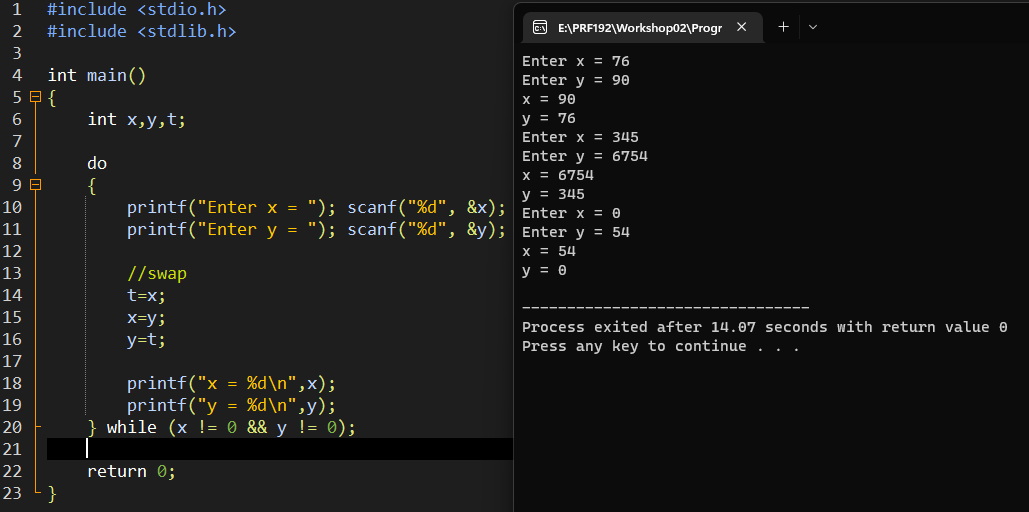
**Program 3 (1 mark)**

|  |  |
| --- | --- |
| **Objectives** | Practice loop statements |
| **Related knowledge** | None |
| **Problem** | Write a C program that will print out **sum** of **integers** inputted from the keyboard until the value 0 is inputted. |
| **Analysis**  *Nouns: sum 🡪 int S;*  *Accepted integral value 🡪 int x* | **Suggested algorithm****(logical order of verbs)**  Begin  S=0;  Do {  Accept x;  If (x != 0) S = S + x;  }  While (x!=0);  Print out S;  End |



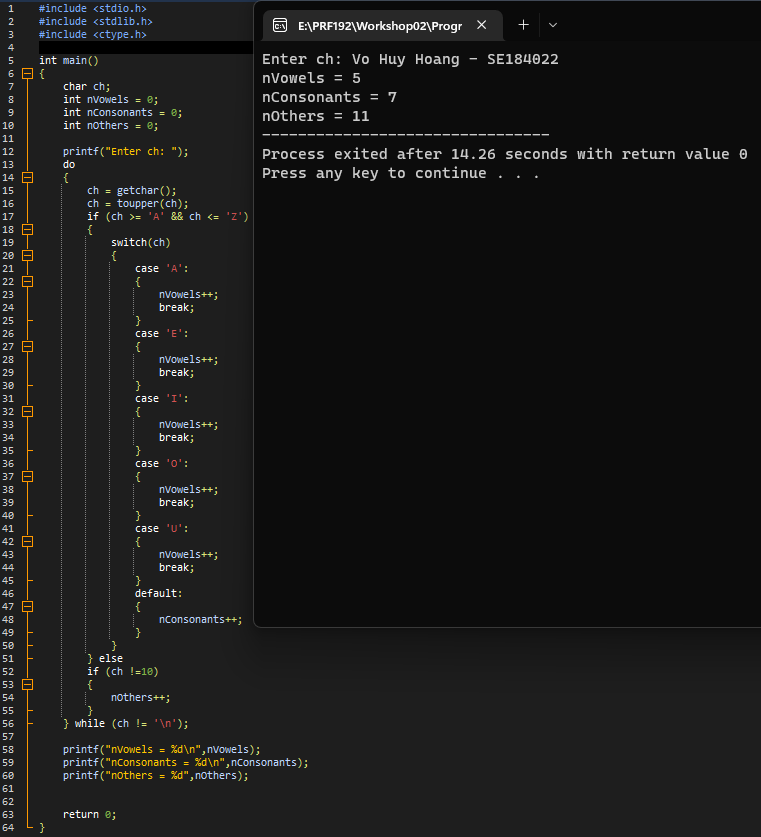
**Program 4 (1 mark)**

|  |  |
| --- | --- |
| **Objectives** | Practice loops statement |
| **Related knowledge** | None |
| **Problem** | Write a C program that will carry out some times: accept two integers, swap these values, print them out to the monitor. The program will terminate when the value of 0 is inputted. |
| **Analysis**  *Nouns:*  *2 integers 🡪 int x, y;* | **Suggested algorithm (logical order of verbs)**  Begin  Do {  Accept x, y;  int t= x; /\* t: temporary variable \*/  x= y;  y= t;  Print out x, y;  }  While ( x!=0 && y!=0);  End |



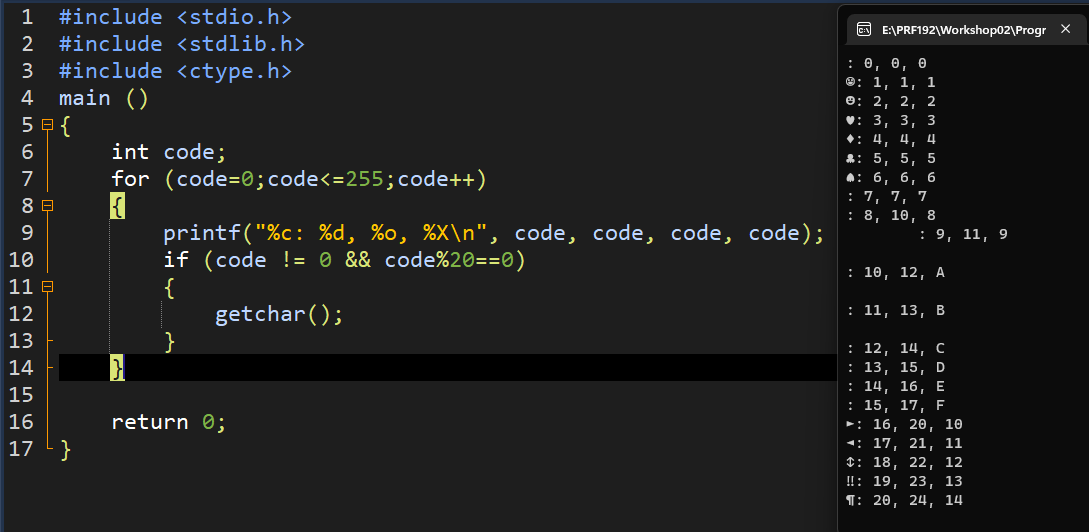
**Program 5: (2 marks)**

|  |  |
| --- | --- |
| **Related knowledge** | Use the function **getchar()** –stdio.h**,** to input a character, the function **toupper(ch)** to convert a character to uppercase - **ctype.h**  ASCII code of the ENTER key: ‘\n’ |
| **Problem** | Write a C program that will:   * permit user inputting a string of characters. The input operation will terminate if the ENTER key is stroked. * print out the number of vowels, number of consonants, and number of others to the monitor. |
| **Analysis**  *Nouns:*  *inputted character*  *🡪 char ch*  *Number of vowels*  *🡪 int nVowels =0;*  *Number of consonants*  *🡪 int consonants =0;*  *Number of other characters 🡪 int nOthers =0;* | **Suggested algorithm (logical order of verbs)**  Begin  Do {  Accept ch; /\* ch= getchar(); \*/  Convert ch to its uppercase /\* ch= toupper(ch); \*/  If (ch>=’A’ && ch <=’Z’) {  switch (ch) {  case ‘A’ :  case ‘E’ :  case ‘I’ :  case ‘O’ :  case ‘U’ : nVowels ++; break;  default: nConsonants++;  }  }  else if (ch!=10) nOthers++;  }  While ( ch != ‘\n’);  Print out nVowels;  Print out nConsonants;  Print out nOthers;  End |



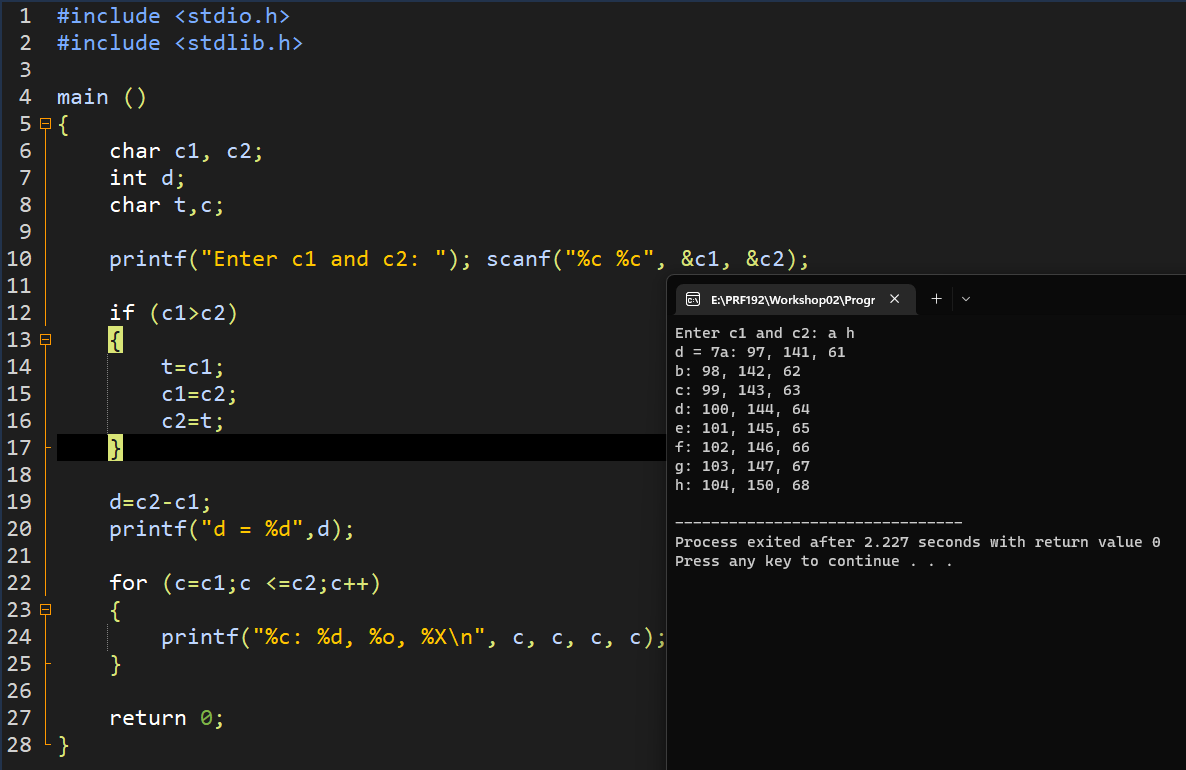
**Program 6: (1 marks)**

|  |  |
| --- | --- |
| **Related knowledge** | Each character will be stored as its ASCII code with value 0..255 |
| **Problem** | Write a C program that will print out the ASCII code table. |
| **Analysis**  ASCII code  🡪 int code | **Suggested algorithm (logical order of verbs)**  Begin  For each code = 0 to 255  { Print out (“%c : %d, %o, %X\n”, code, code, code, code);  If (code !=0 && code %20==0) getchar(); **/\* code page of 20 lines \*/**  }  End. |



**Program 7: (1 marks)**

|  |  |
| --- | --- |
| **Problem** | Write a C program that will accept two characters then print out ASCII code difference between them and characters between them including code values in decimal, octal, hexadecimal expansions in ascending order. |
| **Analysis**  2 character  🡪 char c1, c2  Difference   * 🡪 int d;   Character for swapping operation   * 🡪 char t   Character for looping   * Char c | **Suggested algorithm (logical order of verbs)**  Begin  Accept c1 ;;  Accept c2;  If (c1 > c2 )  { t = c1; c1 = c2; c2= t;  }  d = c2 – c1;  Print out d;  For each c from c1 to c2  { Print out (“%c : %d, %o, %X\n”, c, c, c, c);  }  End. |



**END**