hochschule hof
University of Applied Sciences

Applications of Object Oriented Concepts by using C++

(Concept and Tools)

Under the guidance of Dr. Prof. Martin Thost

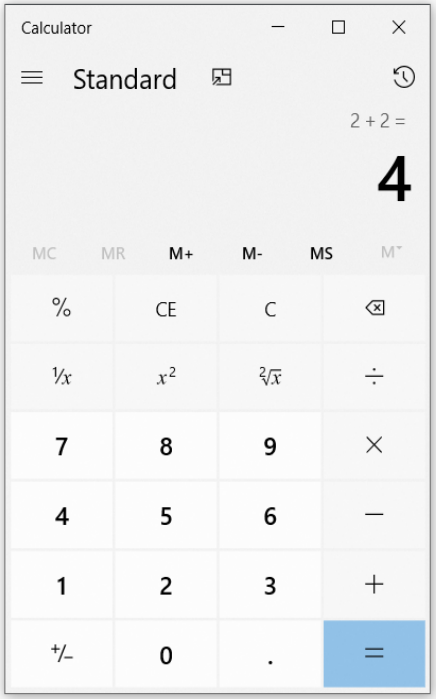
By:
Sukanta Maity (00610919)
Subarna Mitra (00540819)

Introduction

```
int findSum(int num1,int num2)
{
int sum=num1+num2;
return sum;
}
```

Object1

Object2



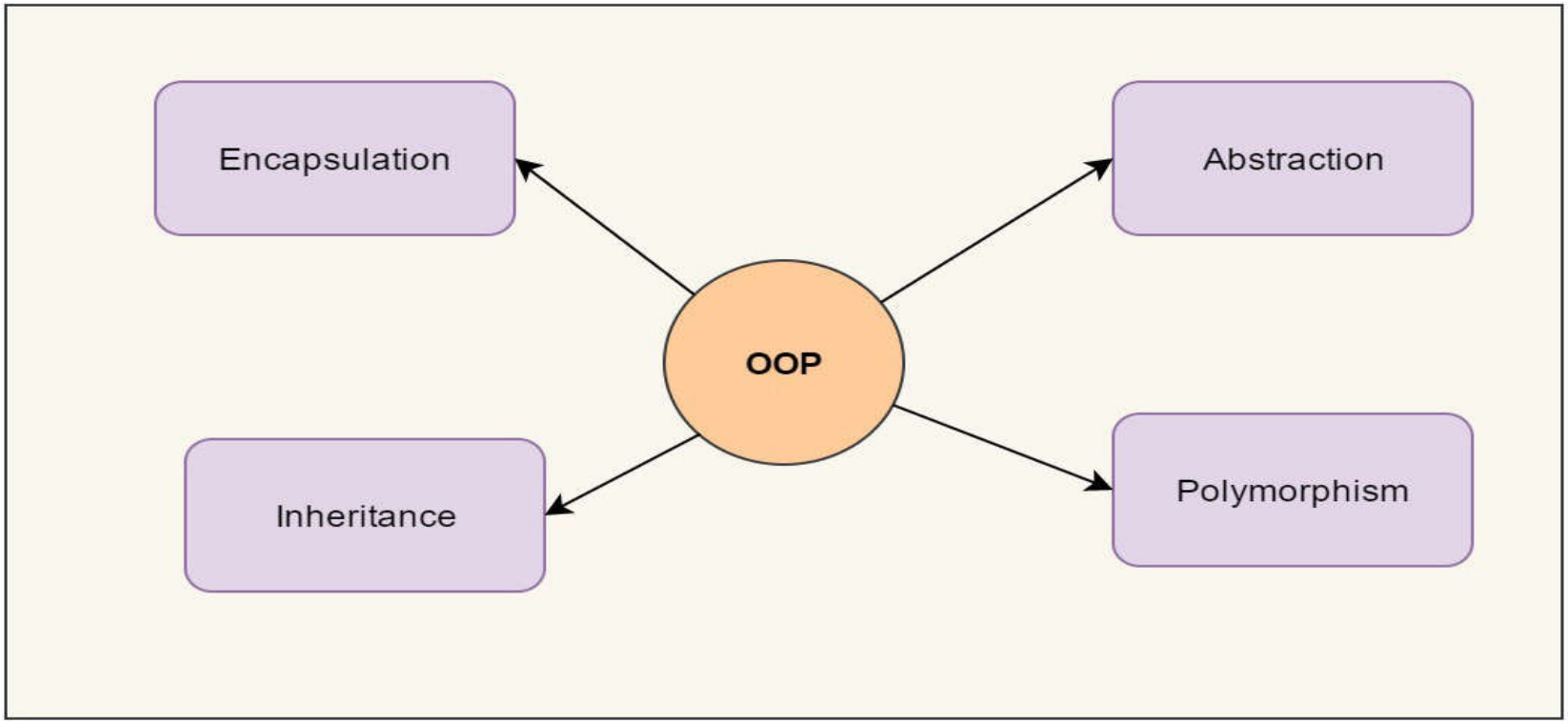
Object3

hochschule hof
University of Applied Sciences

Agenda

- Four pillars of OOP
- Characteristics of OOP
- About inventor of C++
- The root of C++
- Evolution of C++
- Features adopted from Procedural Programming in C++
- Advanced Features of C++
- IDE used for modern C++
- A popular IDE installation and execution process of C++ Program
- Why do Software Developers use C++
- A small project for Desktop Application by Using C++
- Summary
- Questionnaire

Four pillars of OOP



Four Pillars of Object Oriented Programming

Characteristics of OOP

- Emphasis on data rather than procedure
- Division of programs into objects
- Data structures characterize the objects
- Eeparate on the data of an object by using funtions
- Data hiding and cannot be accessed by external functions
- Communication of objects through functions
- Easy way to add new data and functions
- Bottom-up approach

About inventor of C++

- Developed by **Bjarne Stroustrup**
- Extension of the C language
- The designer and original implementer of C++
- Member of the US National Academy of Engineering, and an IEEE, ACM, and CHM fellow.

The root of C++

General-purpose abstraction

Simula

Java

C++

C++11

C++14

C++17

C#

Assembler

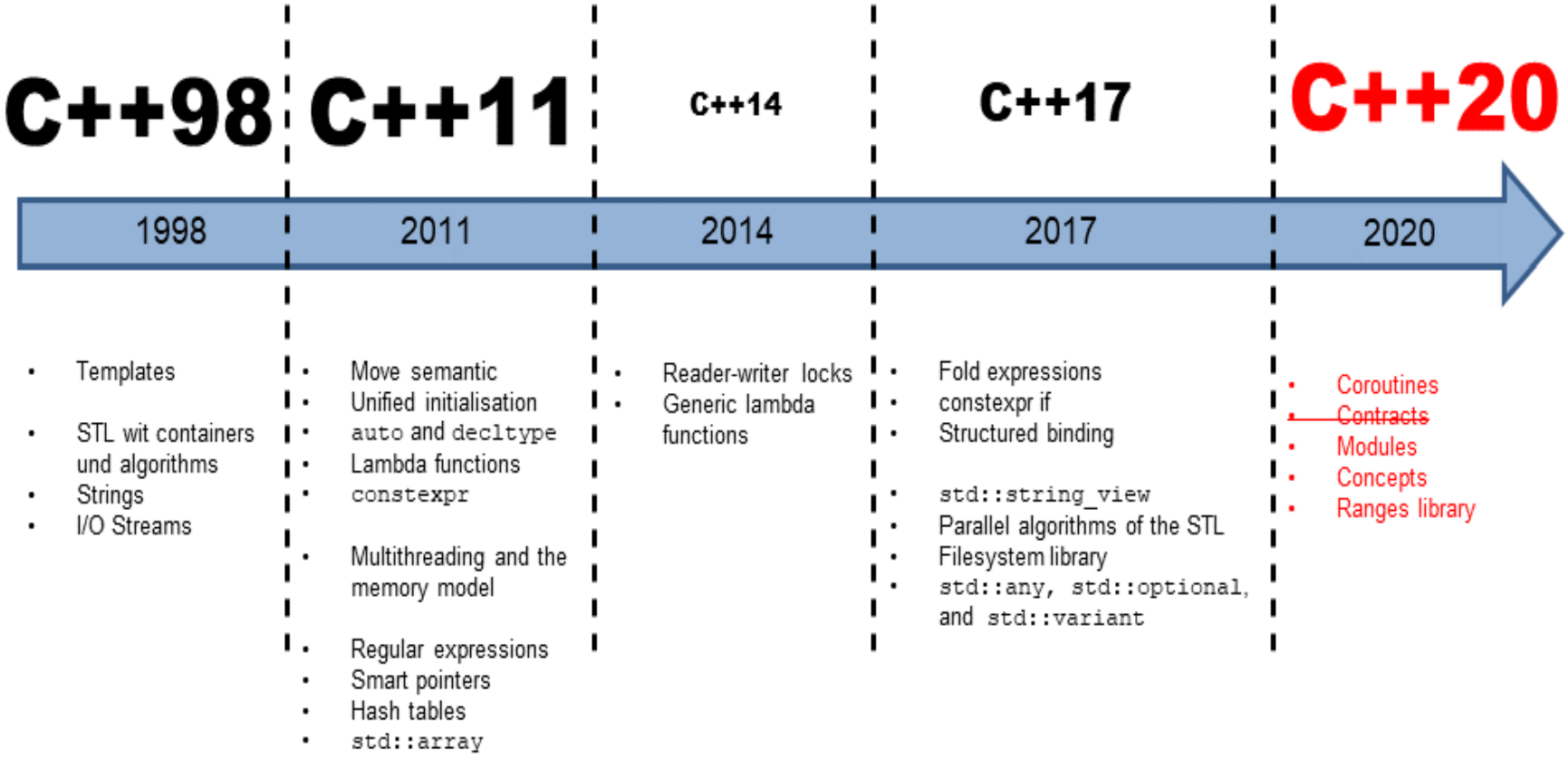
BCPL

C

Basic Combined Programming Language(BPCL)

Evolution of C++

- In 1985, a commercial product.
- In 1998, the published the first international standard for C++ ISO/IEC 14882:1998



Features adopted from Procedural Programming in C++

- Token
- Variables
- Type casting
- Decision making statements
- Loop
- Array
- String handling
- Structure & Union
- Pointer
- File handling

Advanced Features of C++

- Vector
- Virtual functions
- Compiler generated default constructor
- Lambda functions
- Constexpr
- Regular expression
- Random number generator
- Tuple
- Threading
- Container
- Template functions
- Assert

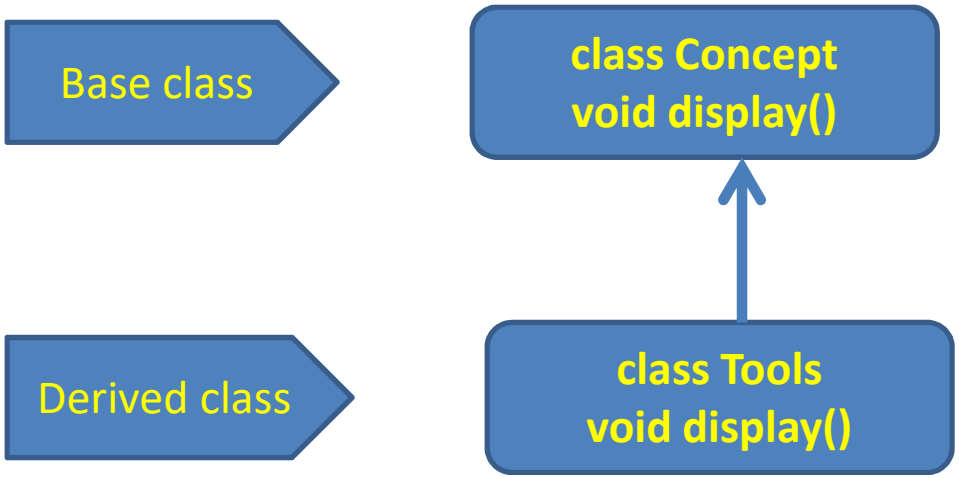
Vector

- Handle dynamic data elements
- Working as sequence containers for stored elements

```
#include <iostream>
#include <string>
#include <vector>
 int main()
 {
   std::vector < std::string > conceptTools;
   conceptTools.push_back("first");
   conceptTools.push_back("sec");
   conceptTools.push_back("third");
   conceptTools:pop_back();
}
```

Virtual functions

- Redefined in derived classes
- Referenced through a base class pointer
- Base class pointer invoke derived class
- Redefined functions during runtime



An Example of Virtual Functions

```
class Concept
{
 public:
            virtual void display()
            {
               cout<<"Concepts";
            }
};
class Tools: public Concept
{
 public:
              void display()
            {
               cout<<"Tools";
            }
};
void main()
{
  Concept C;
  Tools *T=&C;
  T->display();
}
```

Lambda Functions

```
#include<iostream.h>
#include<string.h>

using namespace std;

int main()
{
auto add=[](auto x,auto y)(return
x+y);
std::string str1="Concept",
str2="Tools;
cout<<add(2,3);    //5
cout<<add(str1,str2);
//ConceptTools
return 0;
}
```

“constexpr” Expression

hochschule hof
University of Applied Sciences

- Evaluate the value of a function or variable at compile time

```
#include<iostream.h>
using namespace std;

constexp int add(int a,int b)
{
return a+b;
}

int main()
{
int a,b;
cin>>a>>b;
cout<<add(a,b)<<endl;
cout<<add(2,3);
return 0;
}
```

Regular expression

- Specific pattern
- Concise and flexible string to text

```
#include<iostream.h>
#include<regex.h>
using namespace std;

int main()
{
string str;
cin>>str;
regex e("Concept&Tools");
bool match=regex_match(str,e);
//regex_constant::icase
if(match==true)
cout<<"Matched";
else
cout<<"Not matched";
return 0;
}
```

Random Number Generator

- Generate a sequence without any pattern

```
#include<iostream.h>
#include<ctime.h>
#include<random.h>
int main()
{
std::mt19937 generator;
generator.seed(std::time(0));
std::uniform_int_distribution<uint32_t>dice(1,10);
int random=dice(generator);
cout<<random;

random=dice(generator);
cout<<random;
return 0;
}
```

Tuple

- Hold a number of elements
- Different data types for elements
- The elements initialized as arguments

An example of “tuple”

```
#include<iostream>
#include<tuple>
using namespace std;
int main()
{
 typedef tuple<int, char, float> tp;
tp t1(1,'A',1.2);
tp t2{2,'B',2.2};
cout<<get<0>(t1)<<endl;
cout<<get<1>(t1)<<endl;
cout<<get<2>(t1)<<endl;
cout<<tuple_size<tp>::value<<endl;

auto[first, second,third]=t2;
cout<<first<<endl;
cout<<second<<endl;
cout<<third<<endl;
return 0;
}
```

Threading

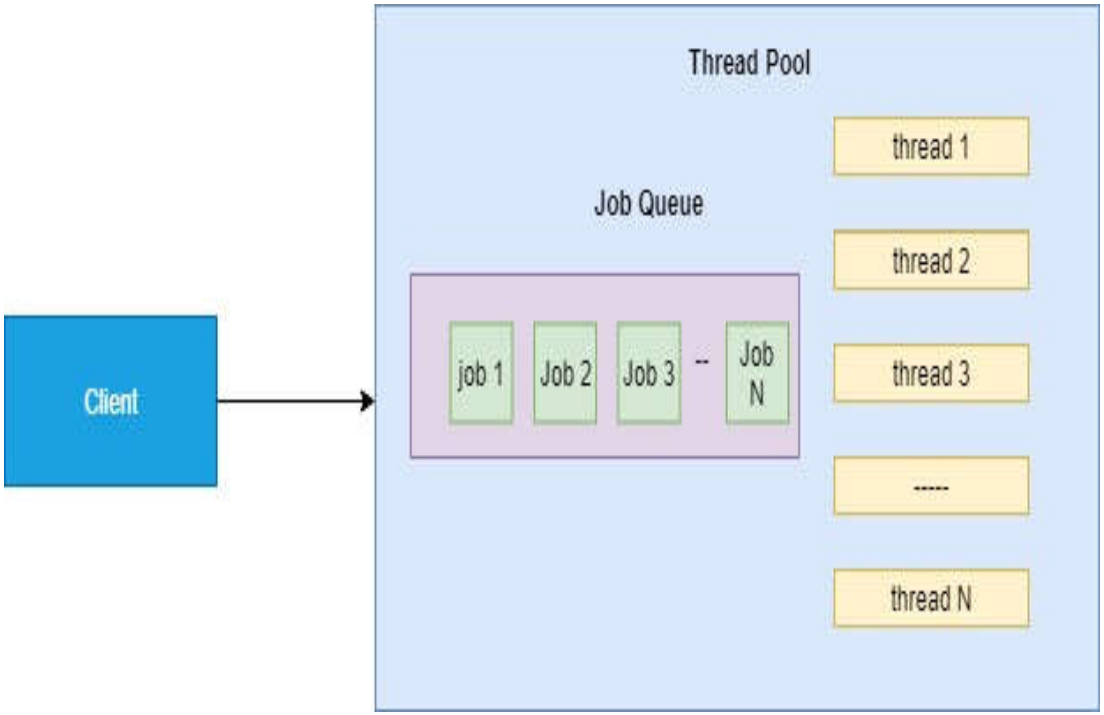
- Light weight process and idea
- Achieved parallelism by dividing a process into multiple threads

```
#include<iostream.h>
#include<thread.h>

using namespace std;

void threadFn(int value)
{
 cout<<"Concepts and Tools";
}

int main()
{
thread t1(threadFn);
t1.join();
return 0;
}
```



Container

- Collection of classes
- Implemented as generic class templates
- Used to hold different kind of objects

Common Containers :-

- Vector
- Queue
- Stack
- Priority_queue
- List
- Set
- Map

An Example of Container “list”

```
#include<iostream>
#include<list>
int main()
{
list<int> list1={1,5,3};
list<int> list2={7,8,2,3};
list1.push_back(10);
list1.sort();
list1.merge(list2);
return 0;
}
```

Template

- Operate with generic type
- Template functions
- Template class

Example of a template function

```
#include<iostream.h>
template<class T>
void show(T a, T b)
{
 cout<"First value   :"<<a;
 cout<"Second value:"<<b;
}
int main()
{
 int a=5, b=10;
 show(a,b);
 //-------------------------
 char *st1="Concept";
 char *st2="Tools";
 show(st1,st2);
 return 0;
}
```

“Assert” Functions

- Tests a program assertion at run time
- Specify expression is false or true

```
#include<iostream>
using namespace std;
void square(int *p)
{
  (*p)=(*p)*(*p);
}
int main()
{
int a=5;
int *p=NULL;
square(p);
cout<<a;
}
```

Without “assert” function

```
#include<iostream>
#include<cassert>
using namespace std;
void square(int *p)
{
  assert(p!=NULL);
  (*p)=(*p)*(*p);
}
int main()
{
int a=5;
int *p=NULL;
square(p);
cout<<a;
}
```

With “assert“ function

IDE used for modern C++

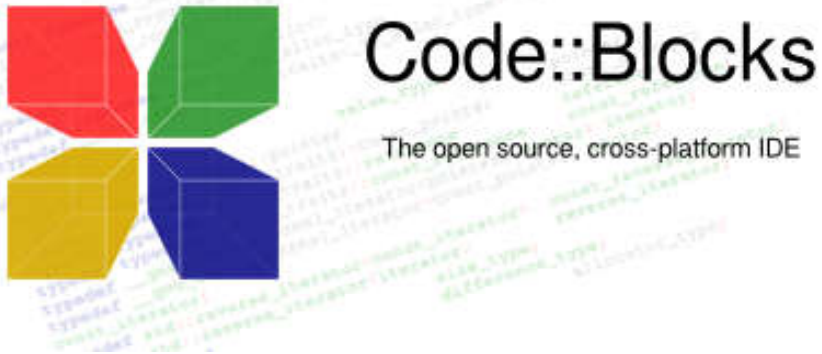
hochschule hof
University of Applied Sciences

- Visual Studio Code 
- Eclipse 
- NetBeans 
- Sublime Text 
- Code::Blocks 
- CodeLite 
- CodeWarrior 

- Dev-C++ 

- Turbo C++ compiler 

A popular IDE installation and execution process of C++ Program



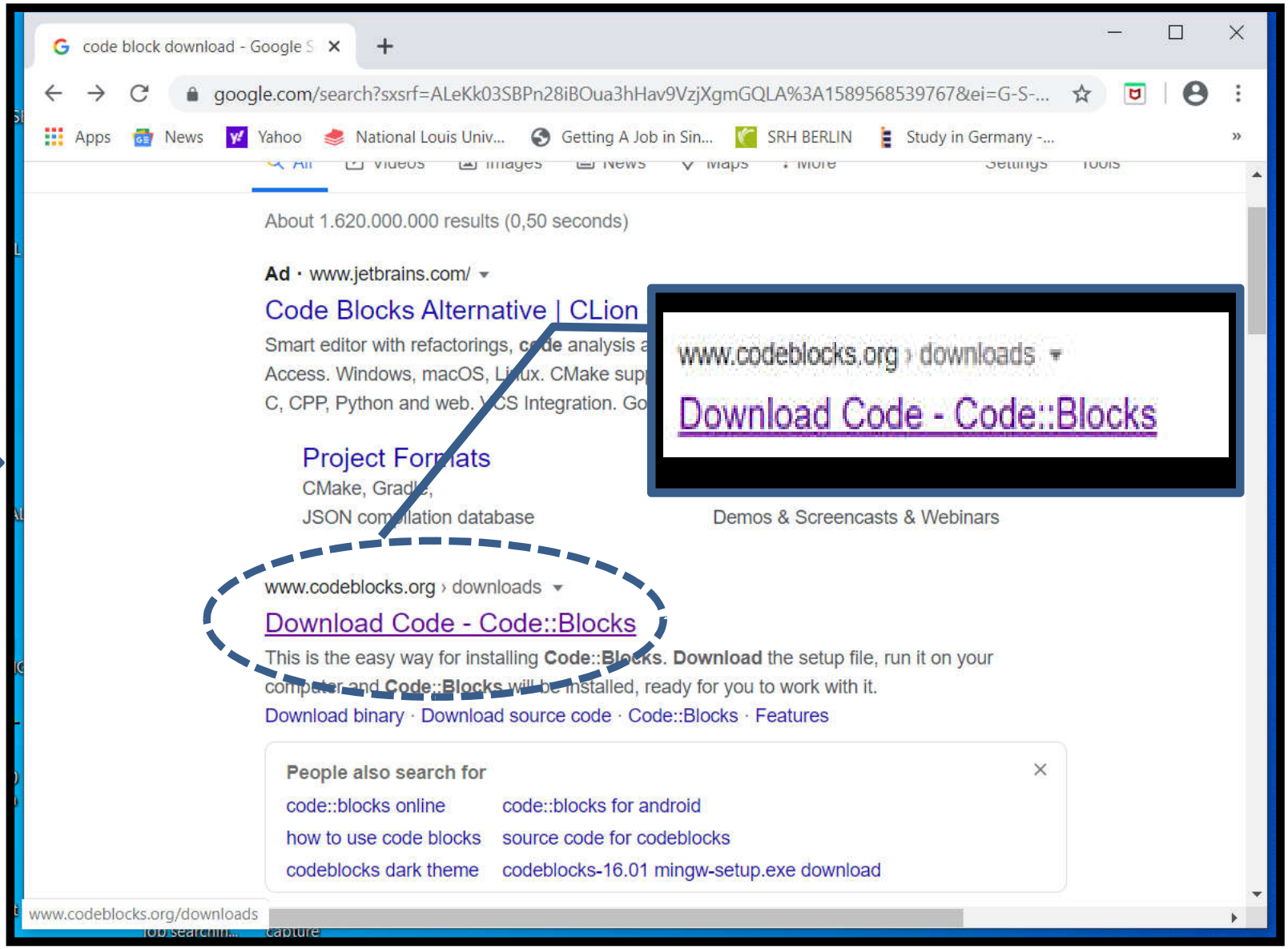
Prerequisite

- Windows 2000/XP/Vista/7)
- Browser
- Internet connection
- 64 bit system

Searching in Google browser

hochschule hof
University of Applied Sciences

Step-1

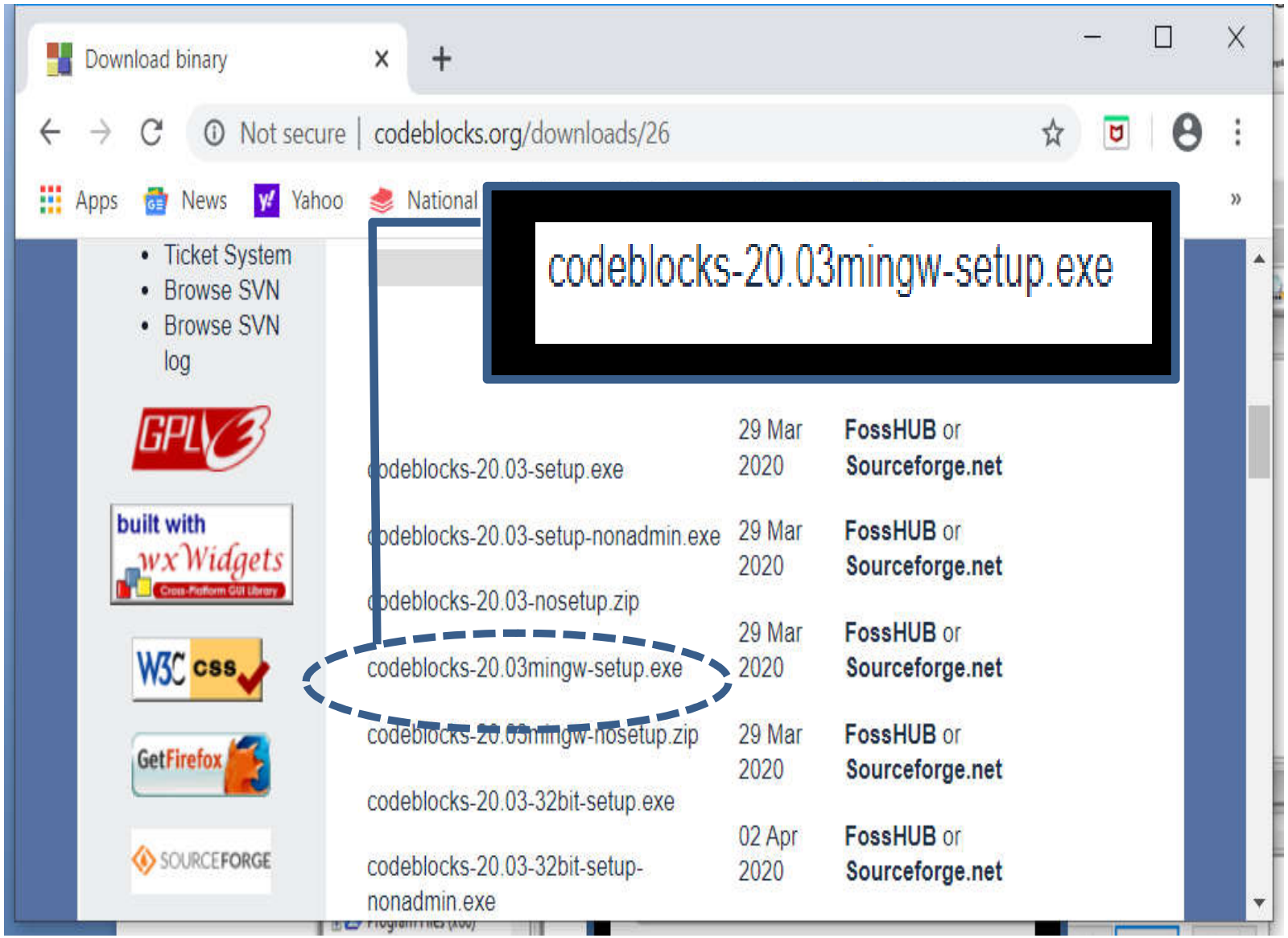


Downloading the release of versions

Step-2

Select the operating system

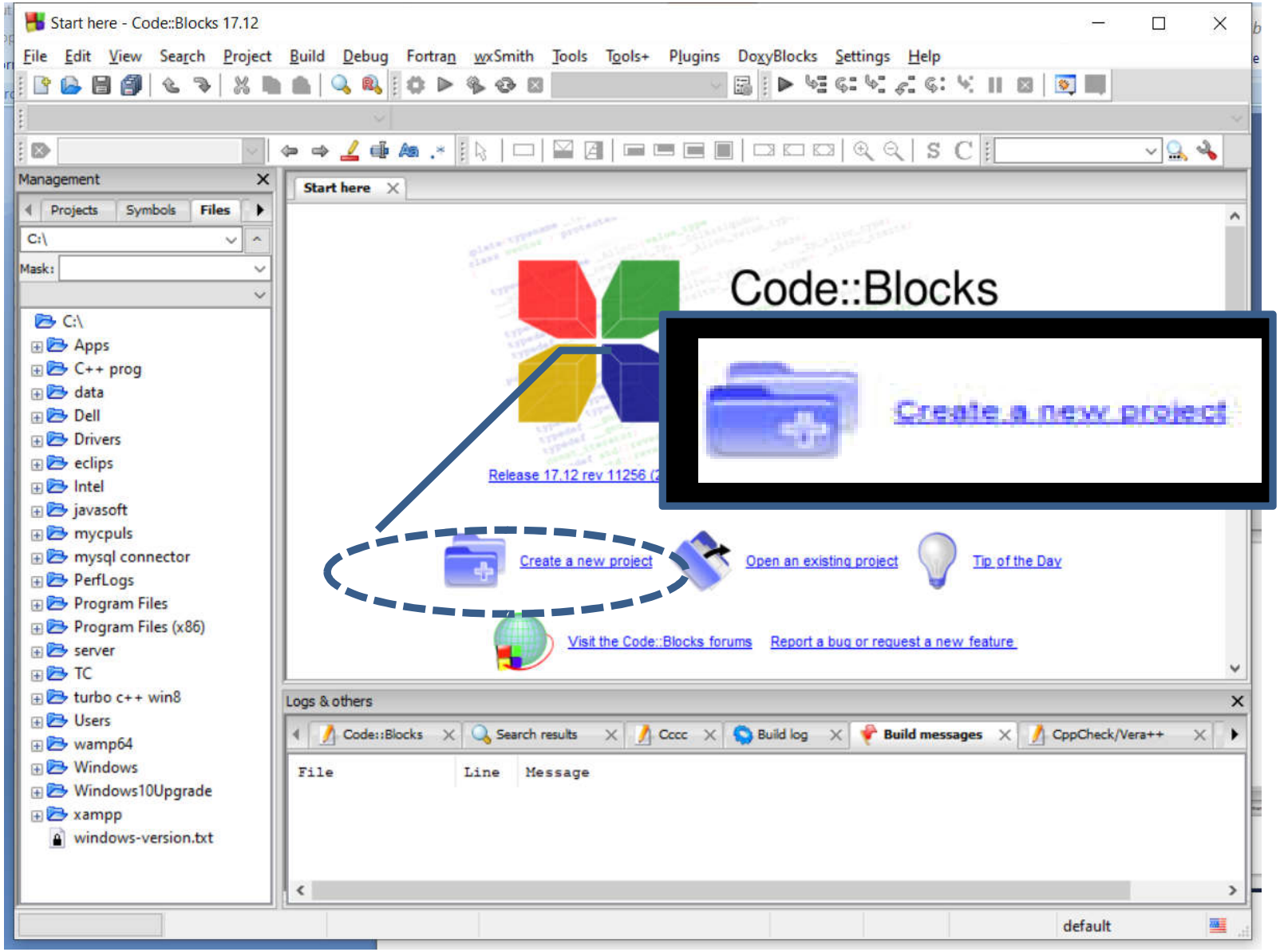
Step-3



Create your first Project

hochschule hof
University of Applied Sciences

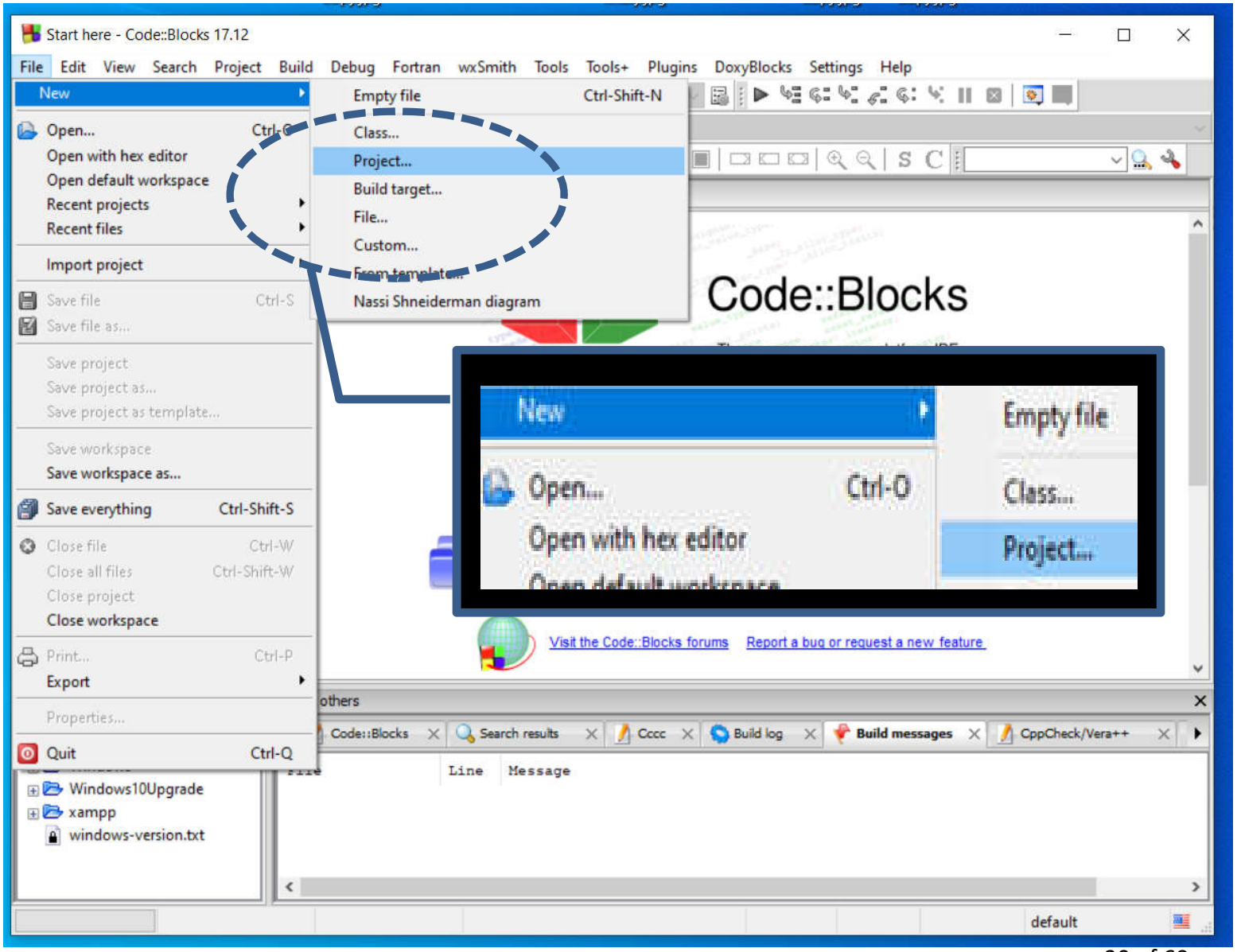
Step-4



Select the menu options

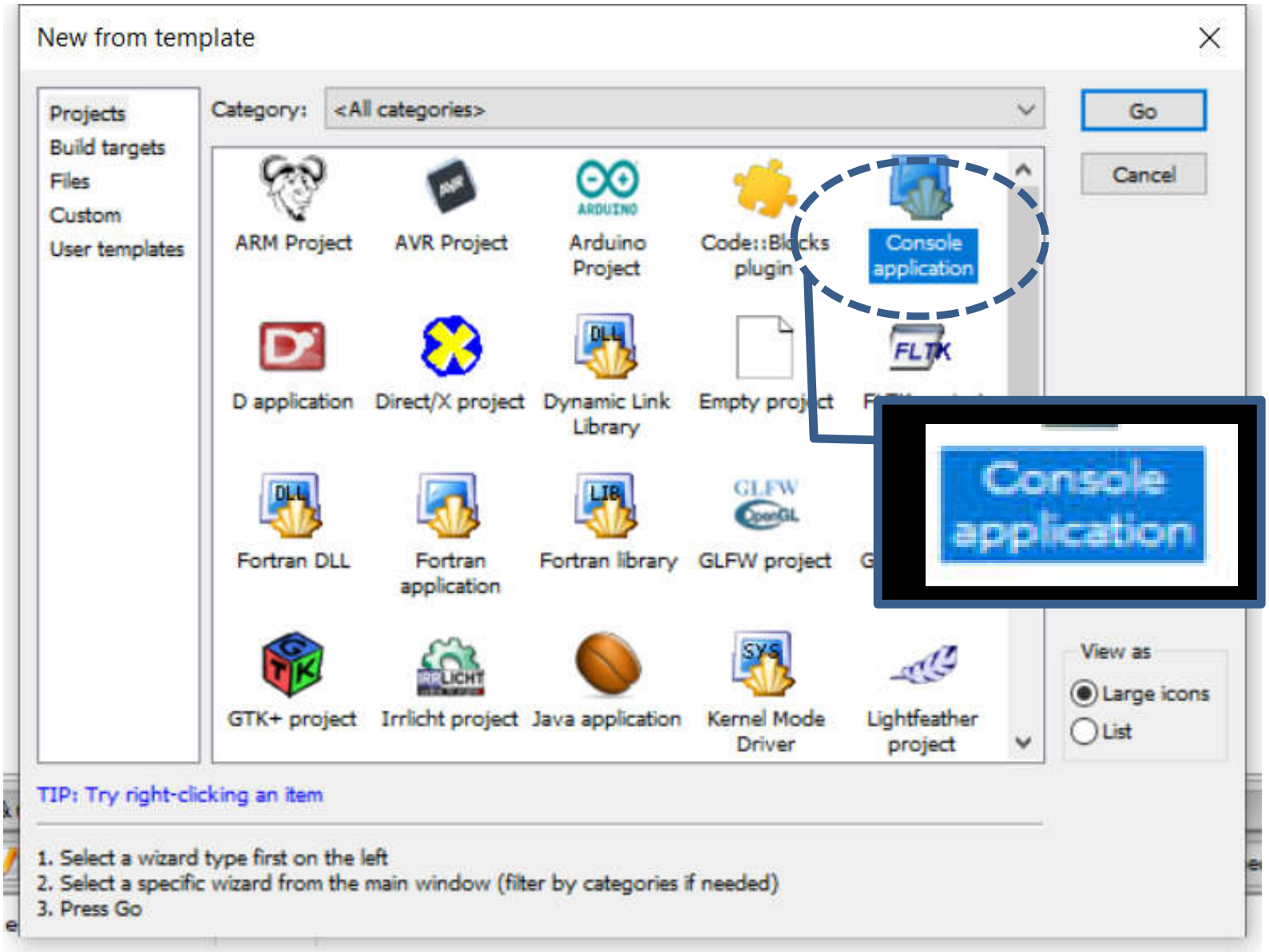
hochschule hof
University of Applied Sciences

Step-5



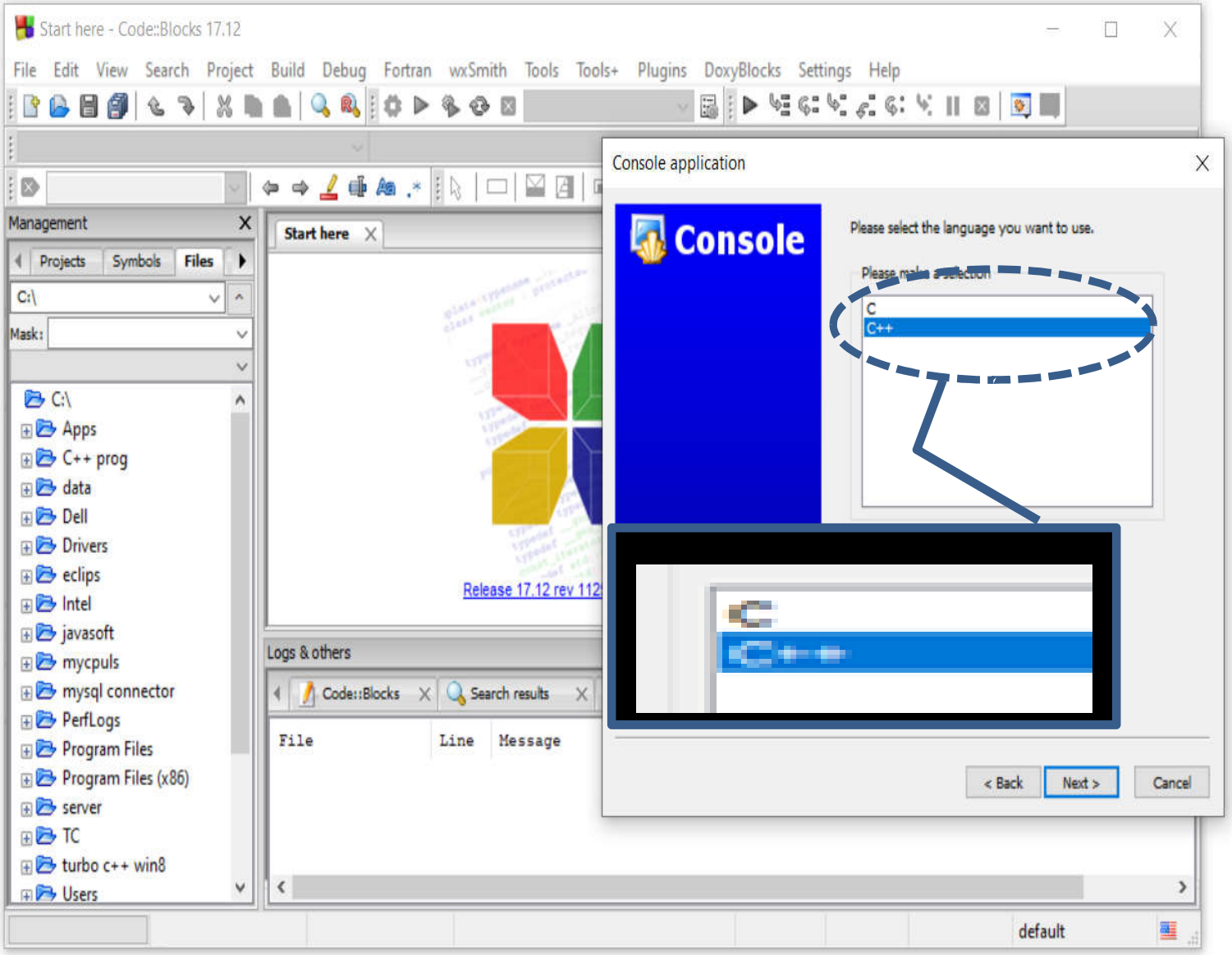
Select the type of applications

Step-6



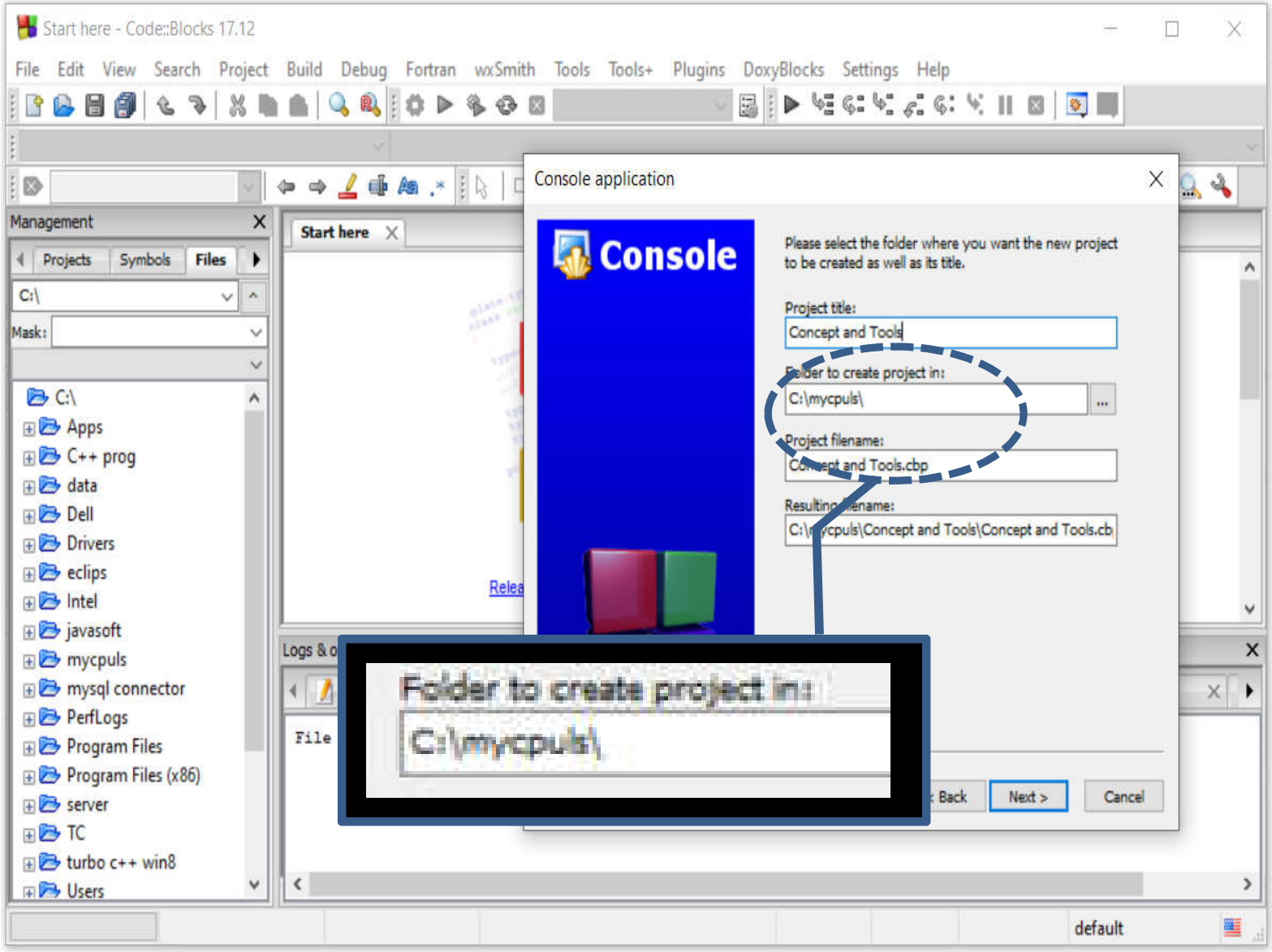
Select the language

Step-7



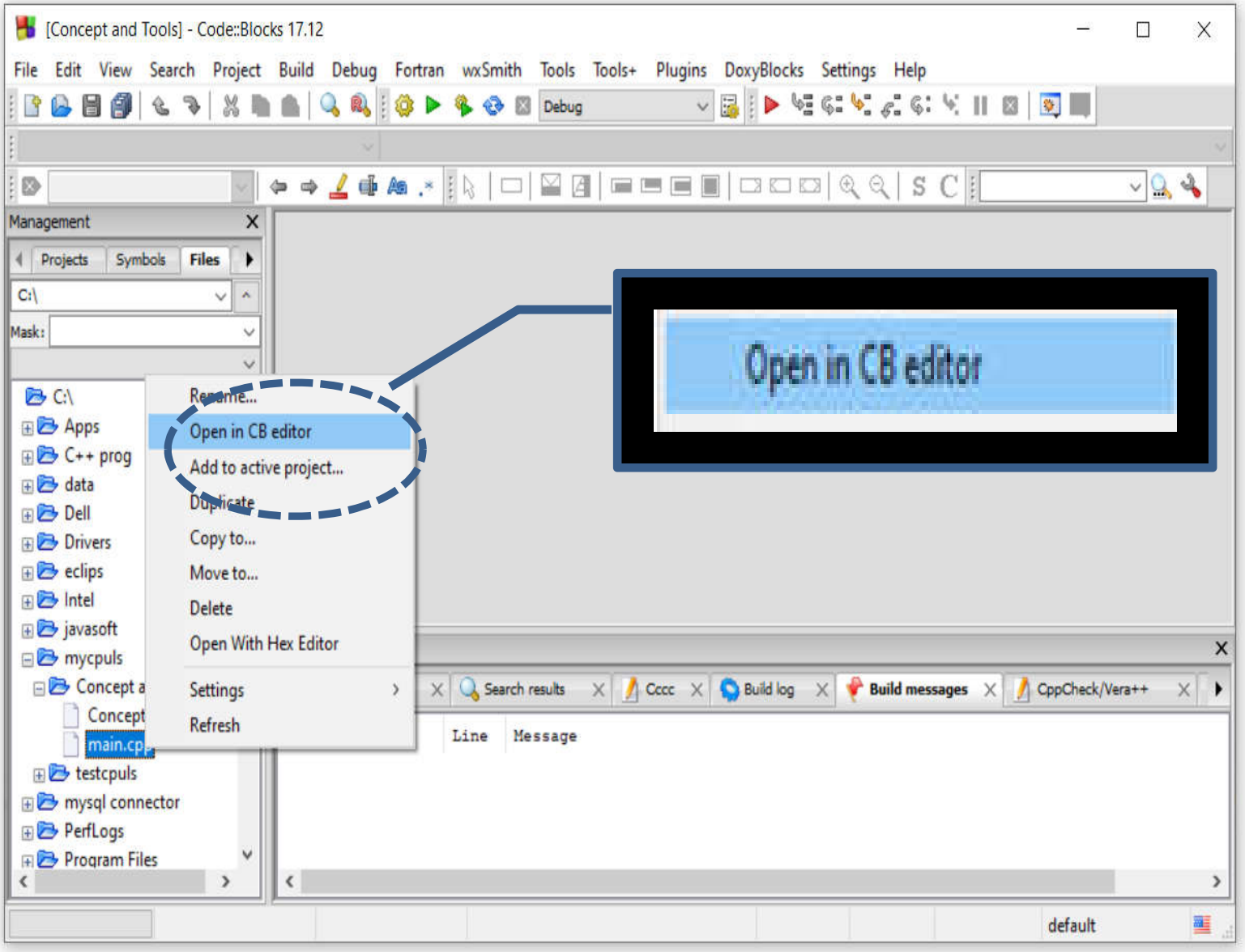
Specify the name of Project

Step-8



Open the Editor

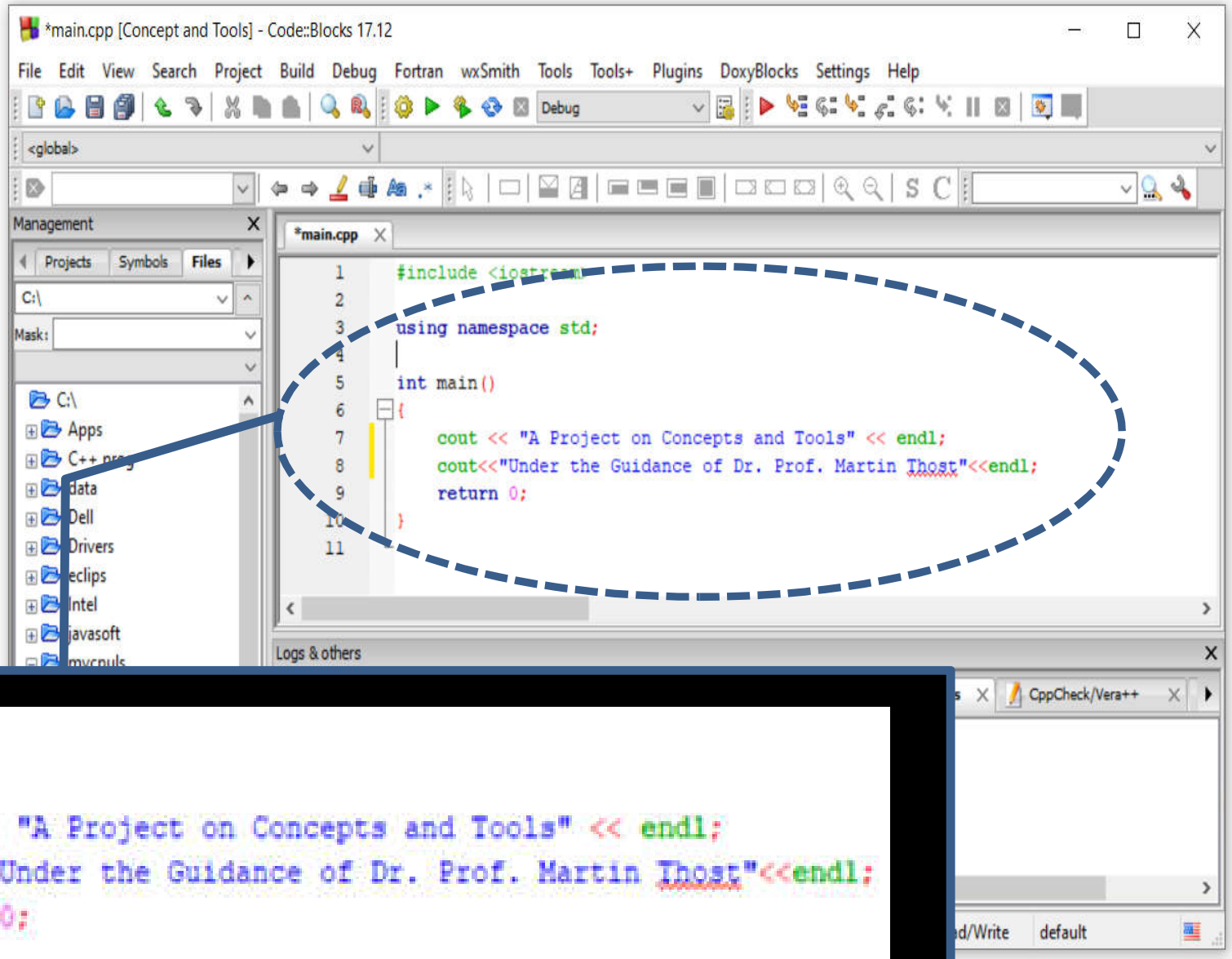
Step-9



Write your code in C++

hochschule hof
University of Applied Sciences

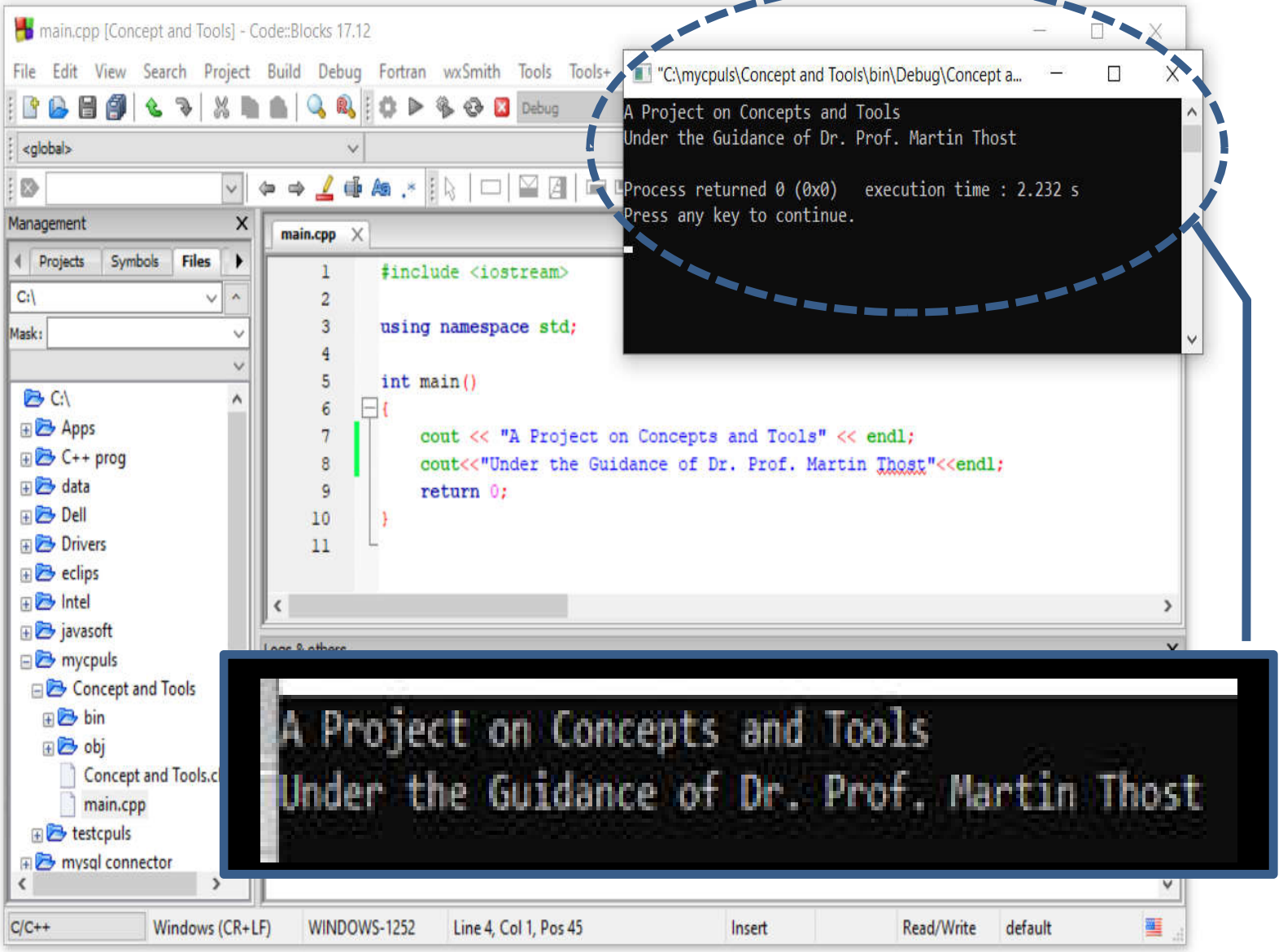
Step-10



Output after execution in Console Window

hochschule hof
University of Applied Sciences

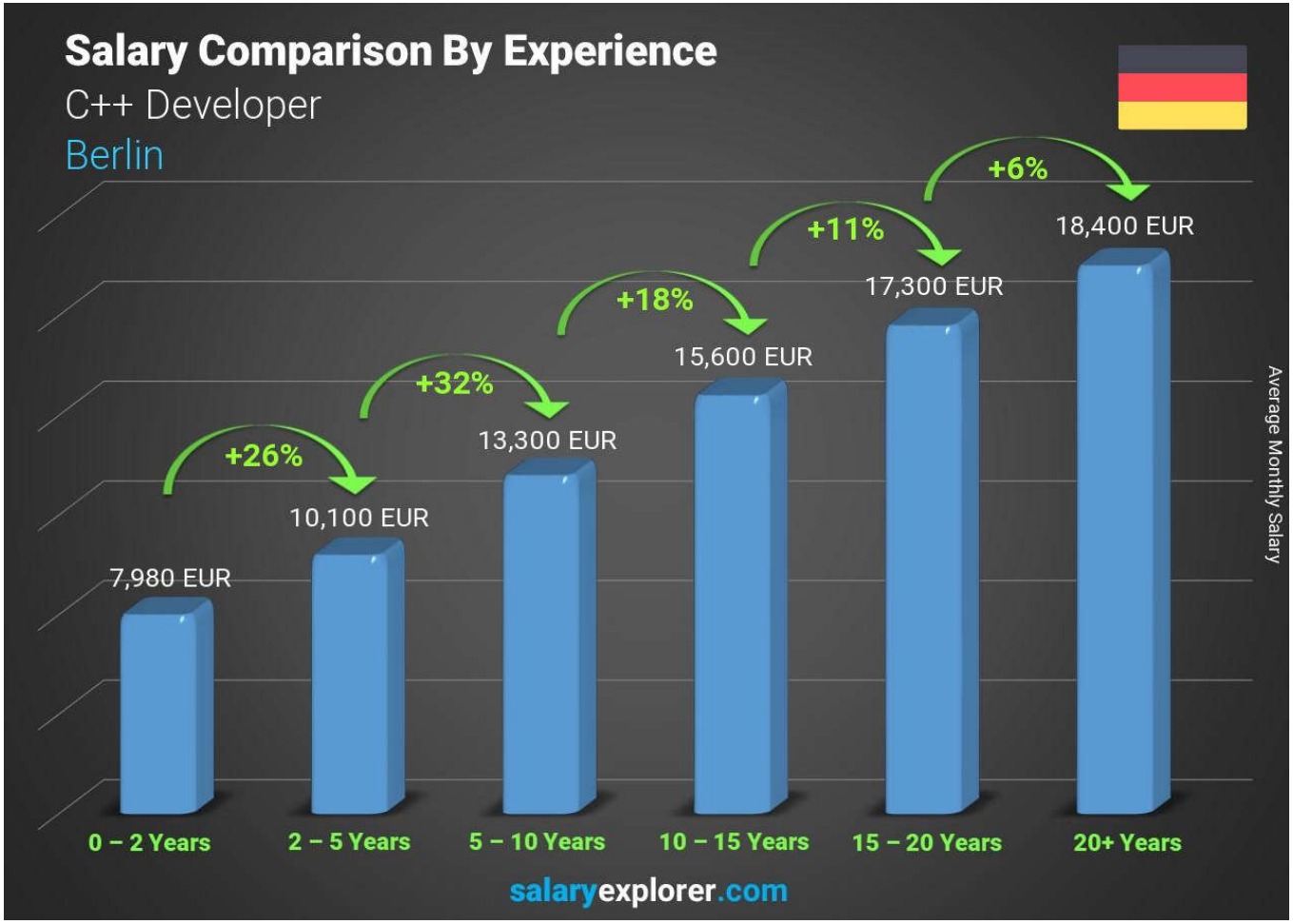
Step-11



Why do Software Developers use C++?

- Stability and evolution
- Tool chains
- Teaching and learning
- Technical community
- Concise expression odd ideas
- Coherence
- Competences
- Compact data structure
- Performance
- Lots of libraries

An Example of growth rate of C++



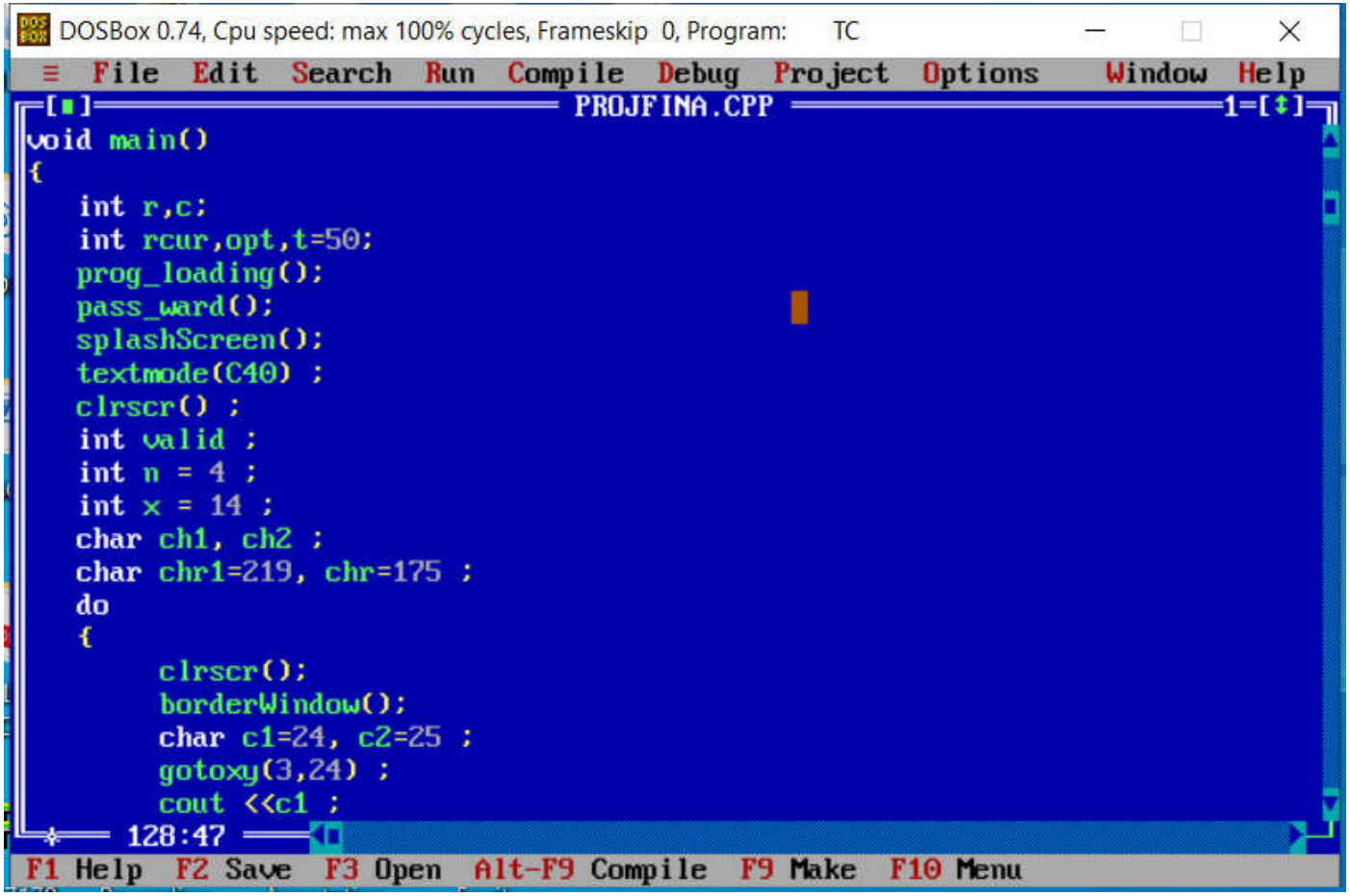
A small project

“STUDENT MANAGEMENT SYSTEM “ for German School

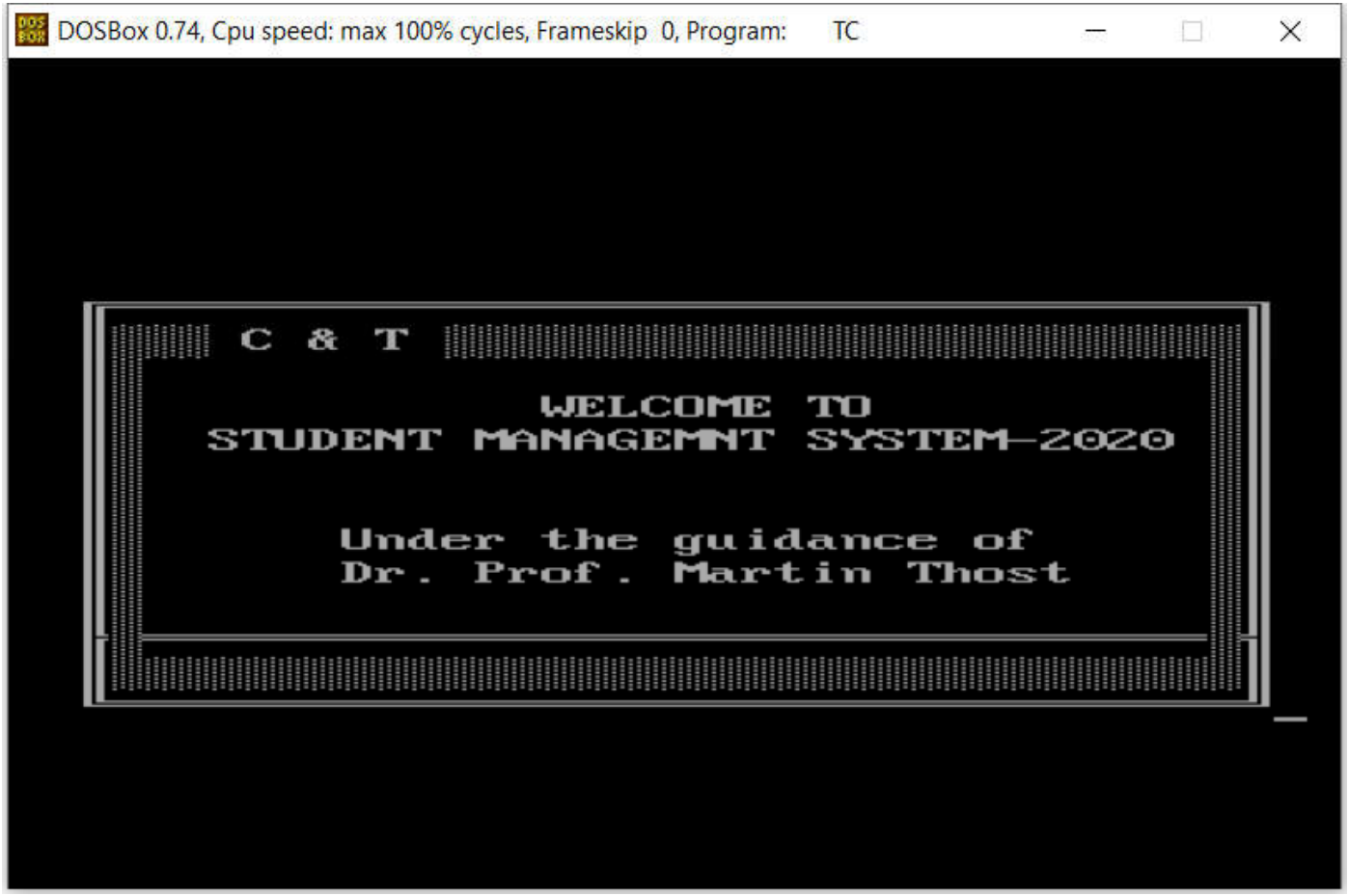
A Desktop Application by Using C++

Demonstration of Coding and Program run

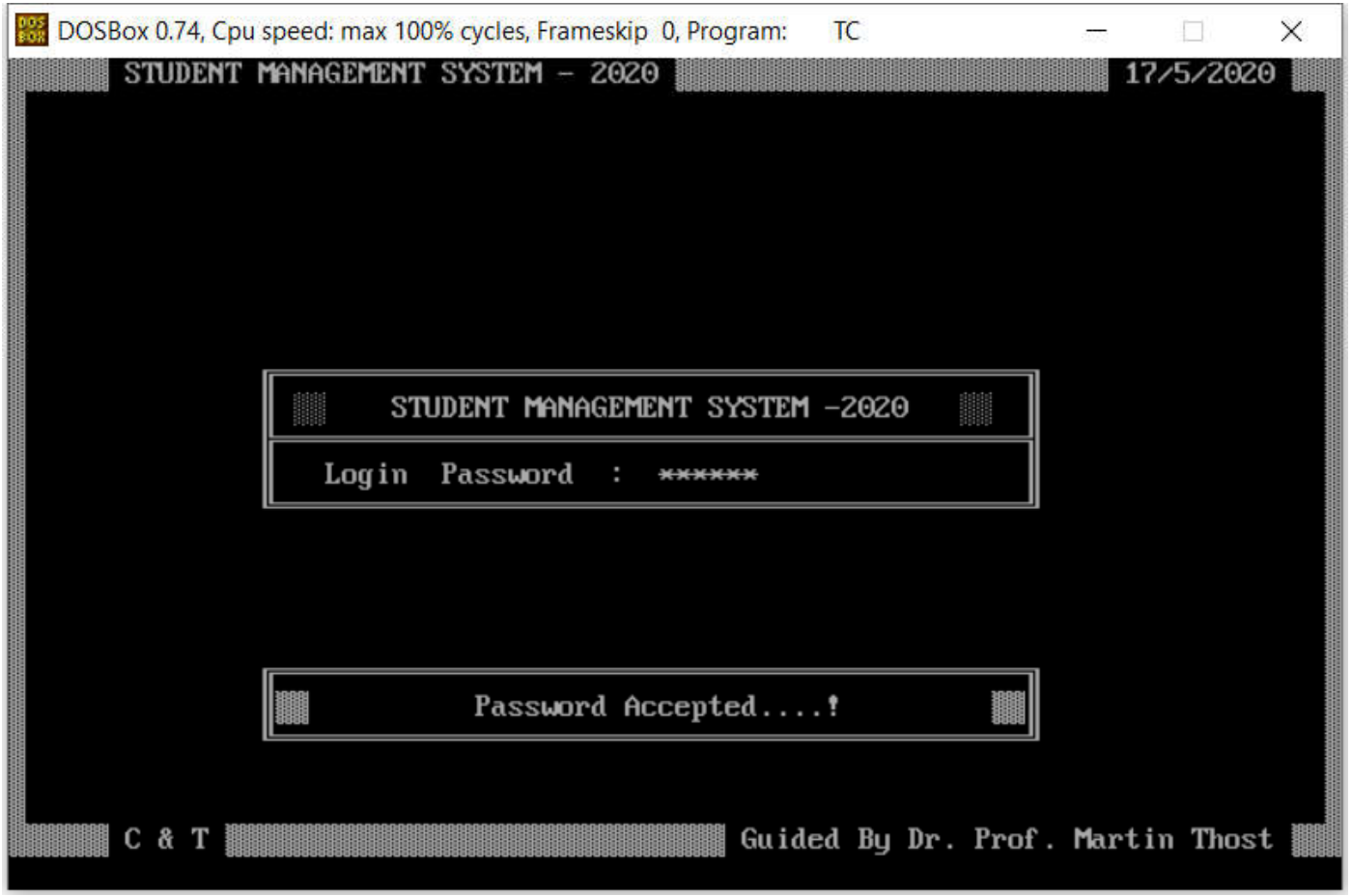
A screenshot of Programming with IDE



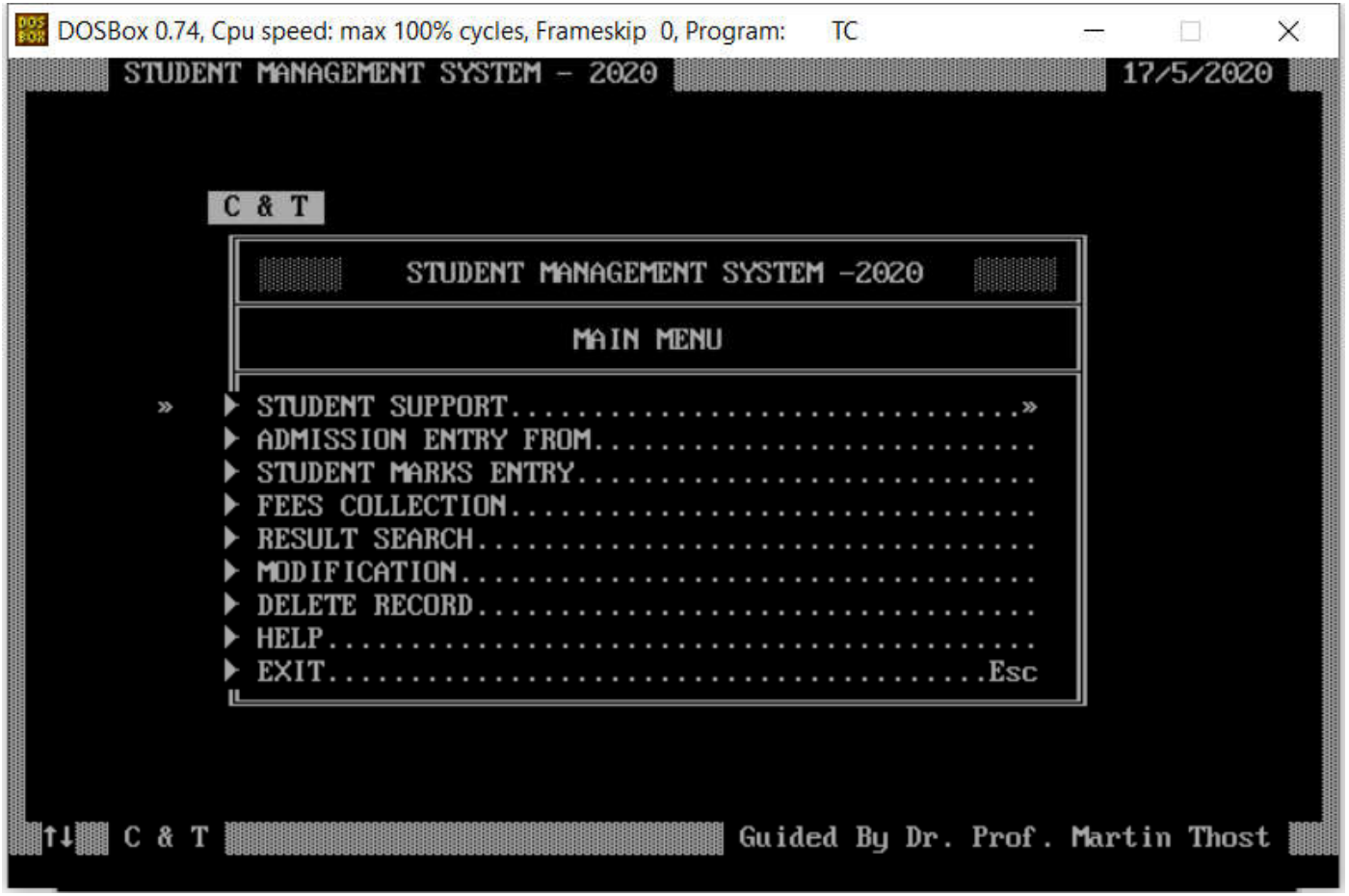
Screenshot of Splash screen (Screenshot-1)



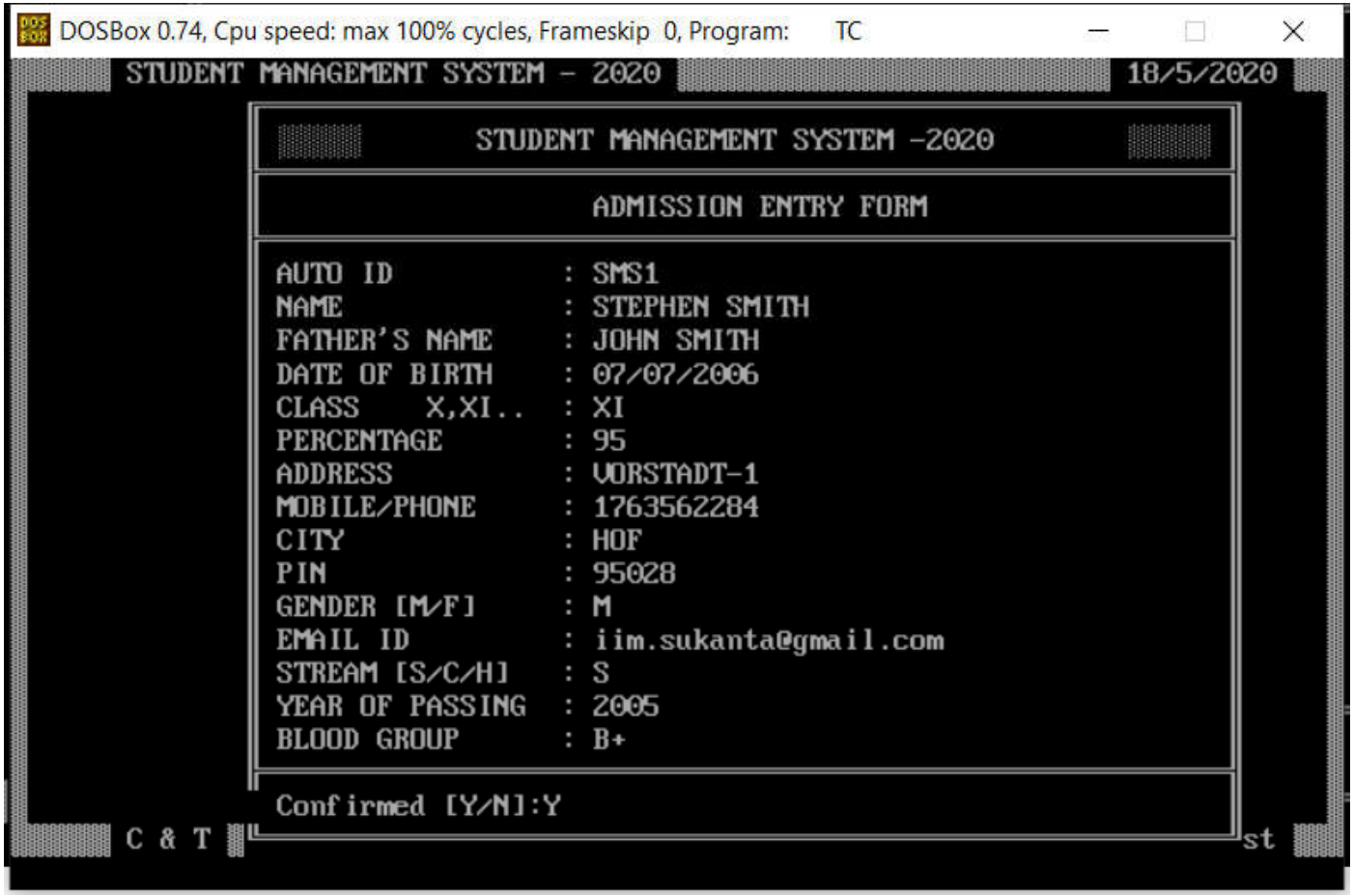
Password Window (Screenshot-2)



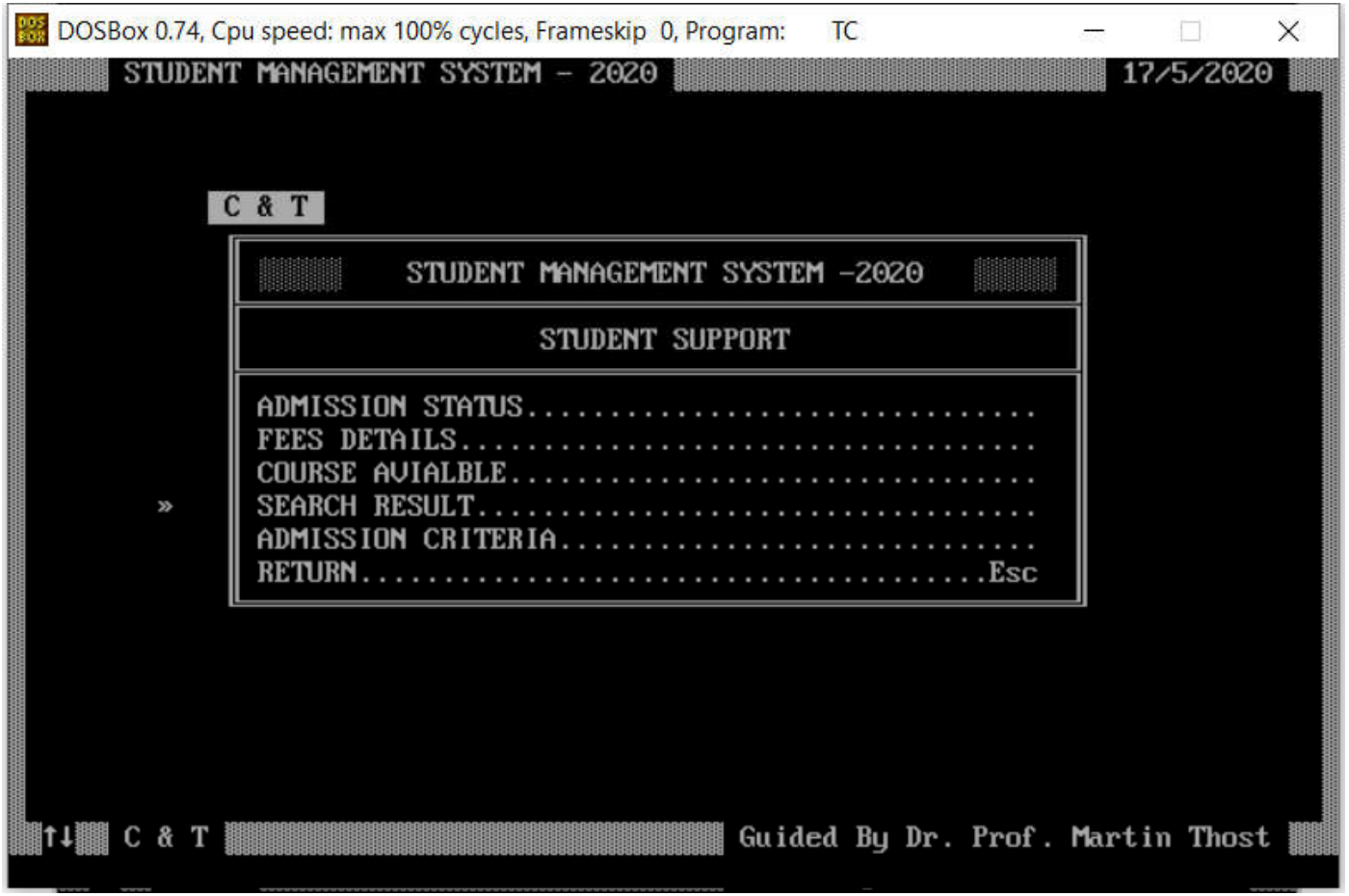
Main menu (Screenshot-3)



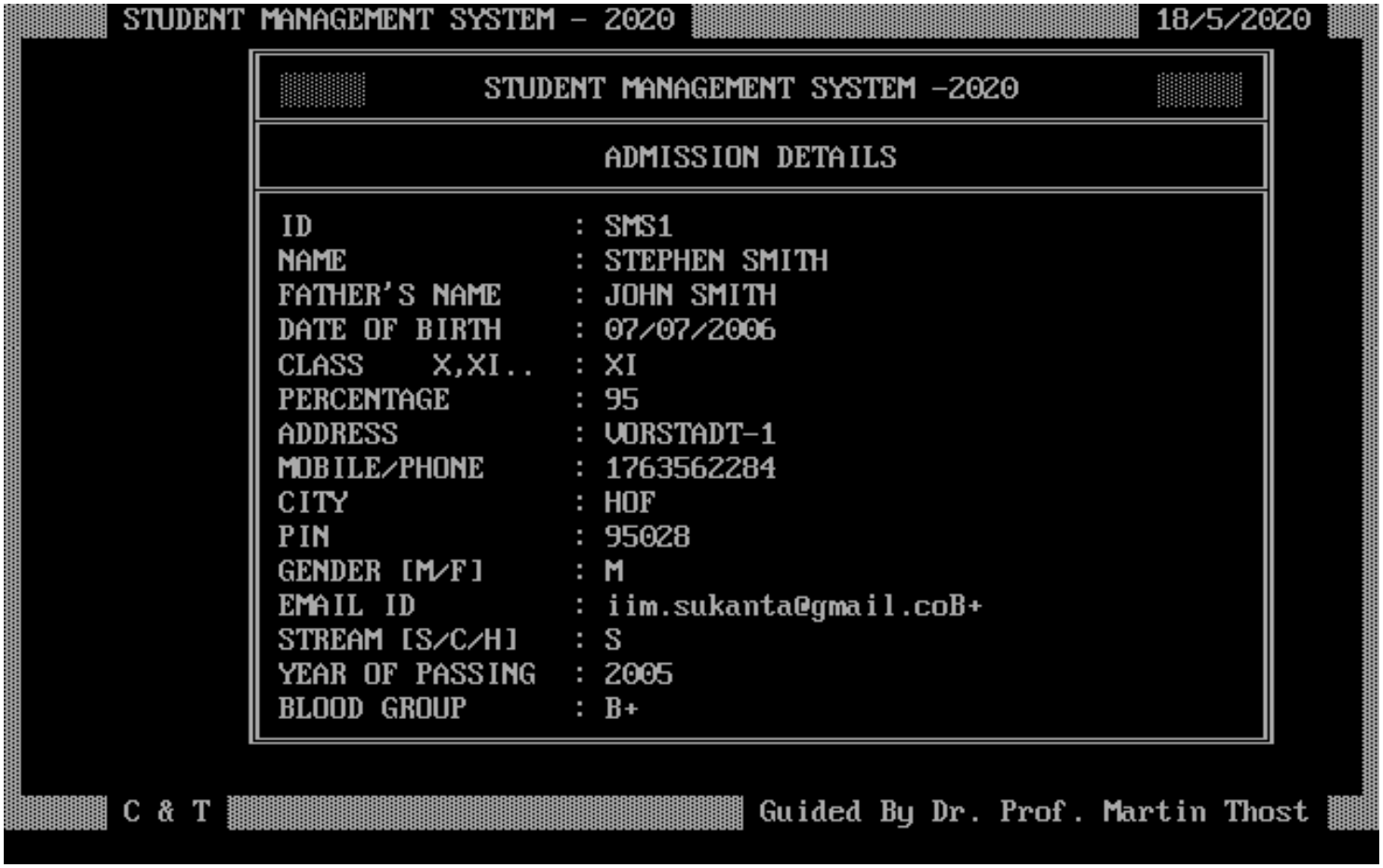
Admission Form (Screenshot-4)



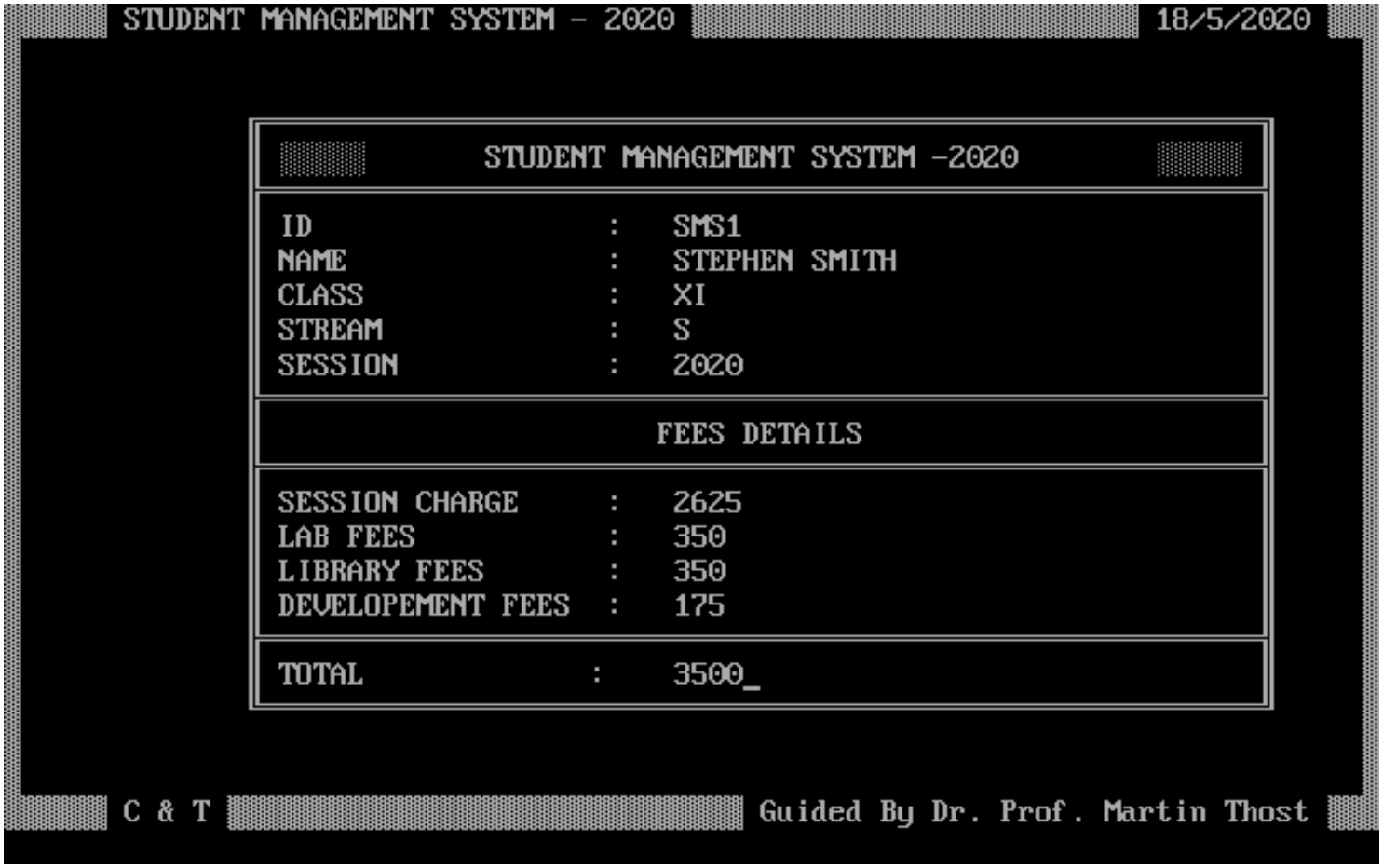
Student support Window (Screenshot-5)



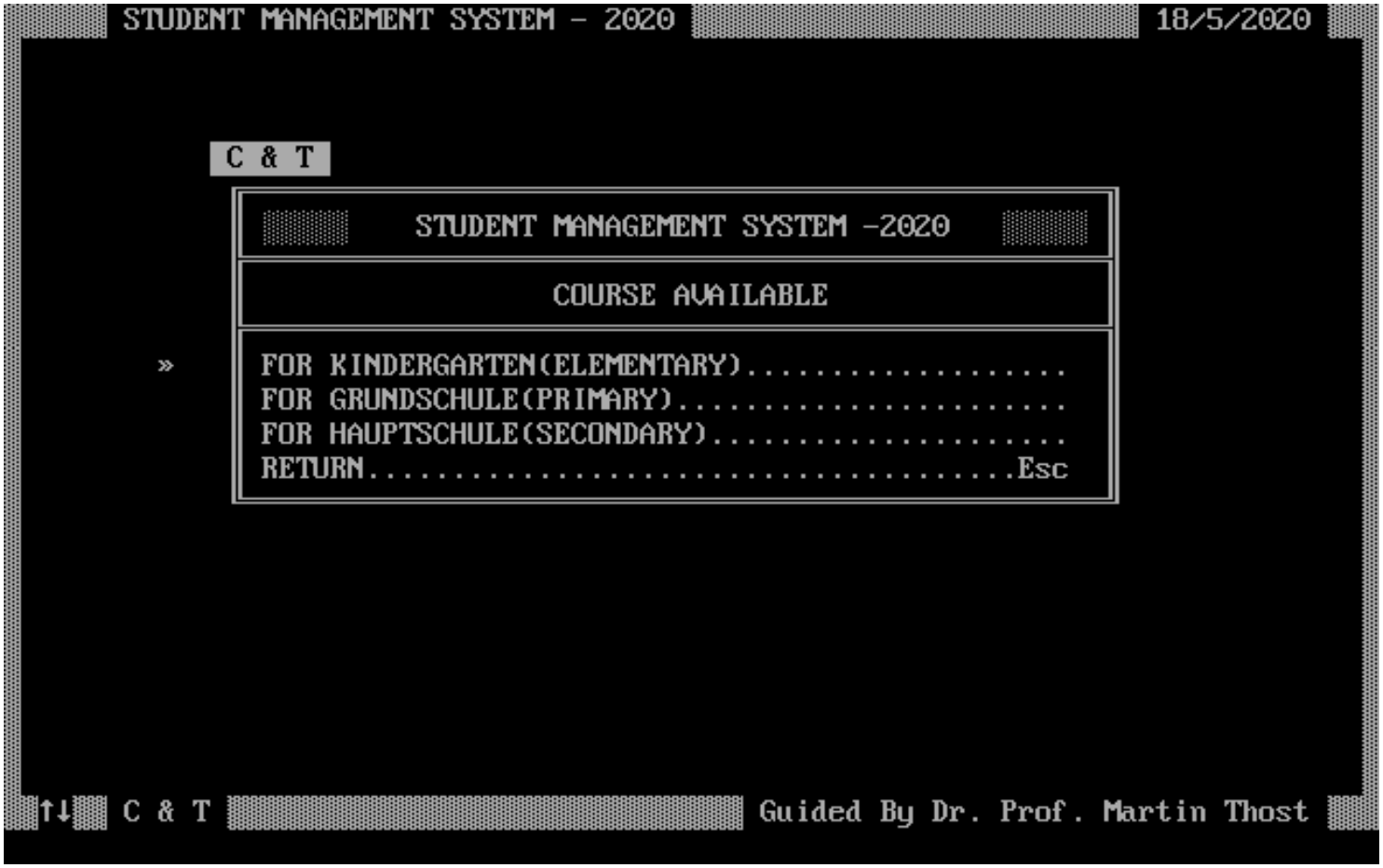
Student Details (Screenshot-6)



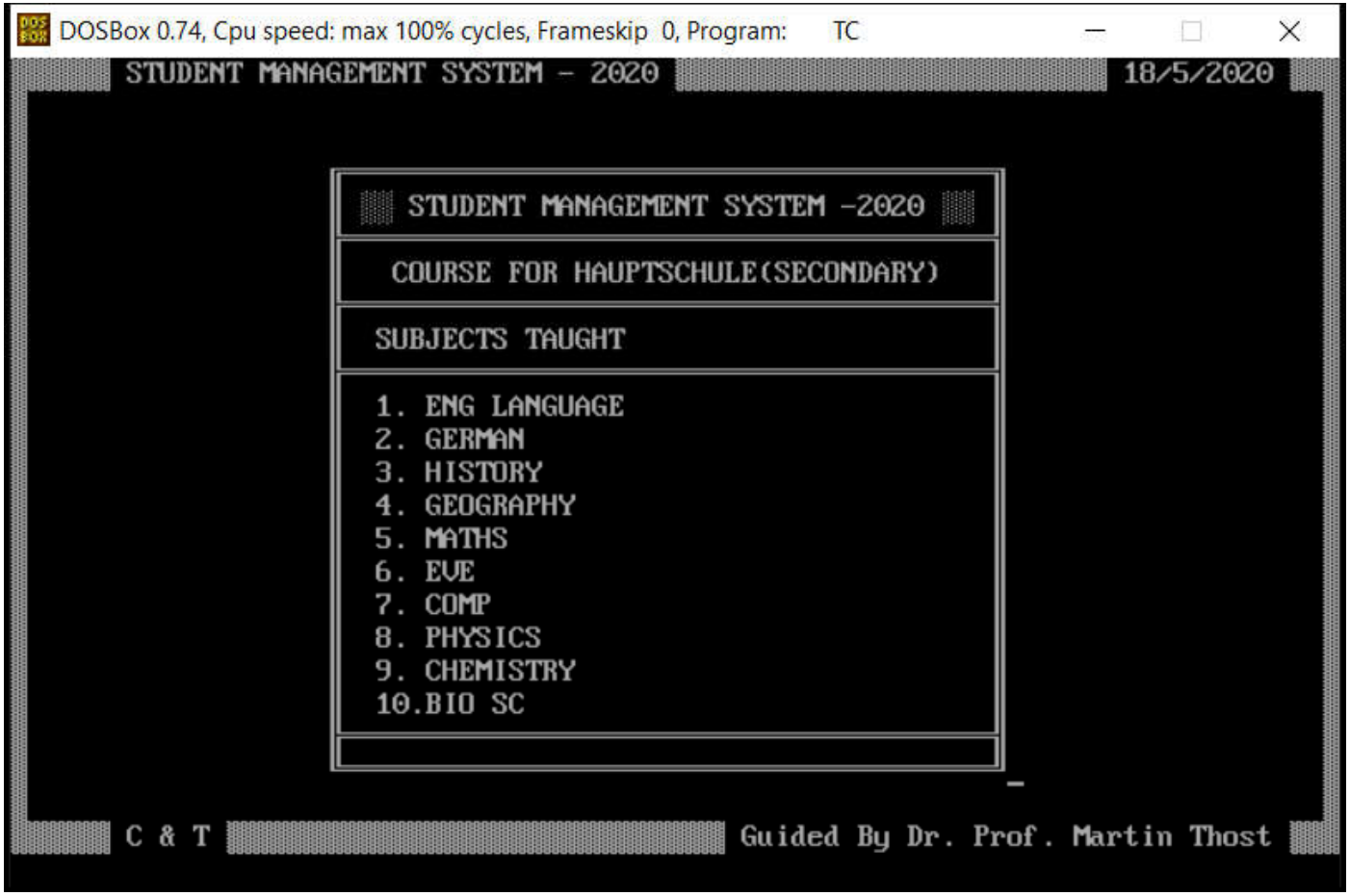
Fees Details (Screenshot-7)



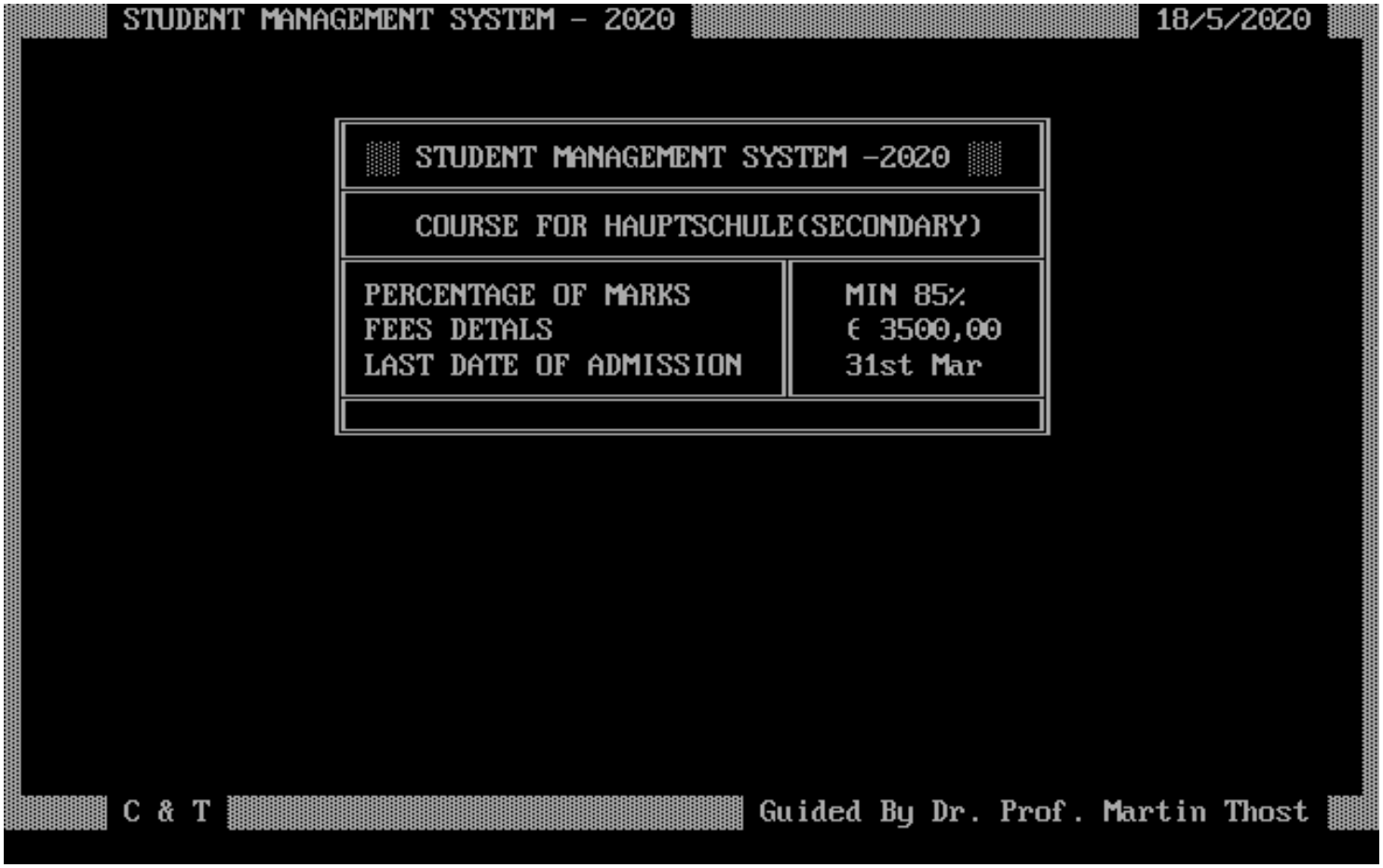
Course Details Option Window (Screenshot-8)



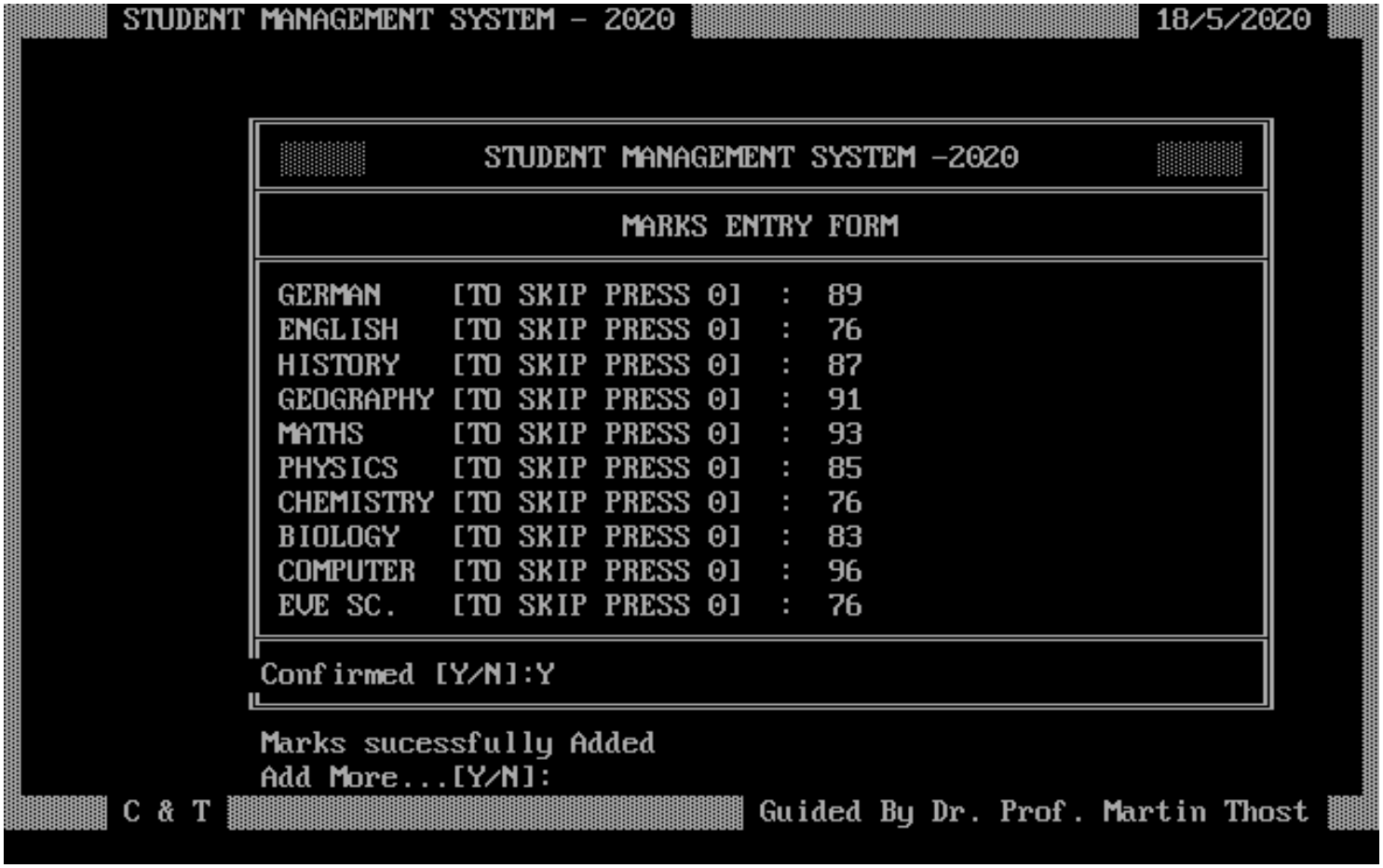
Subjects in Hauptschule (Screenshot-9)



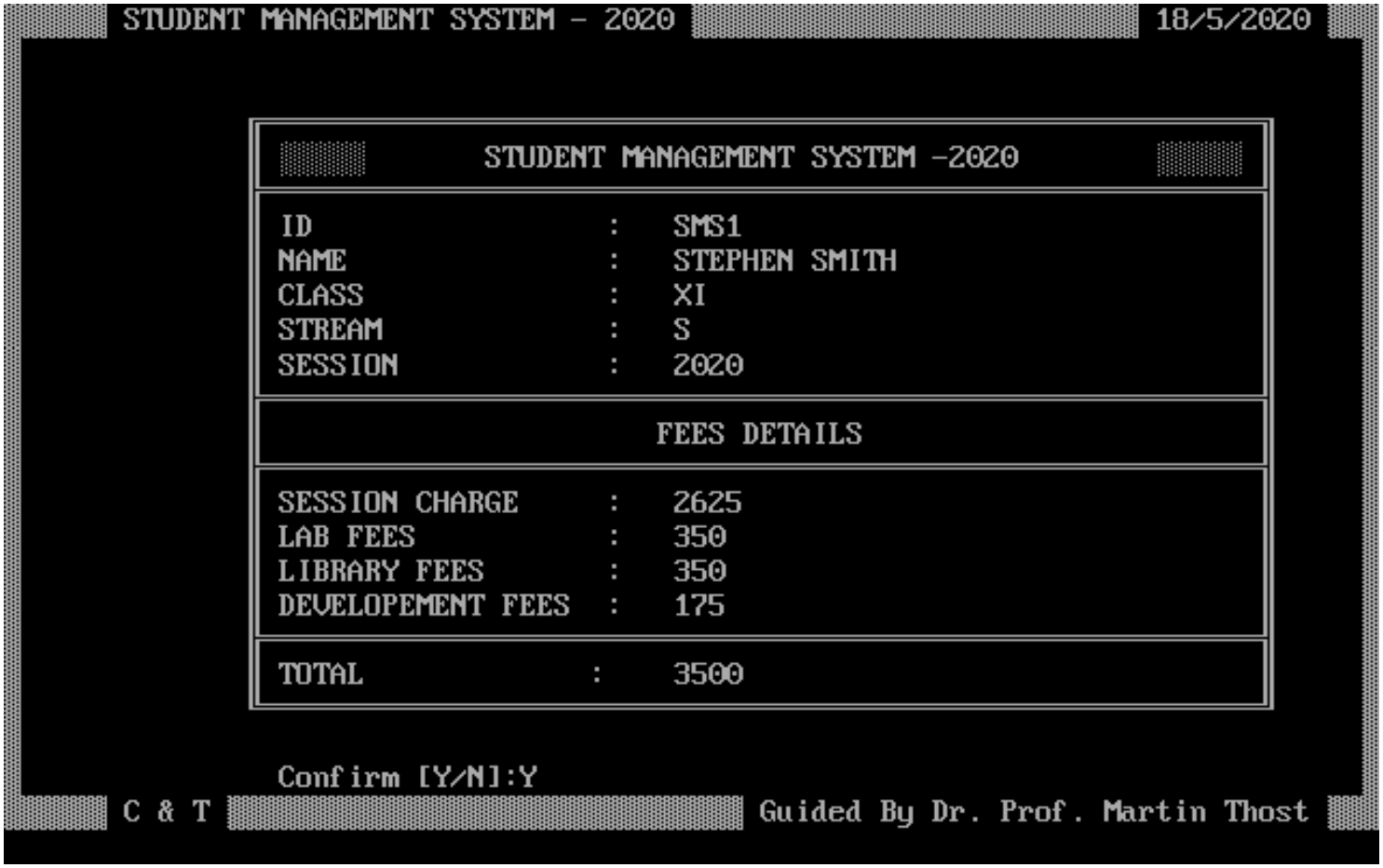
Fees Details for Hauptschule (Screenshot-10)



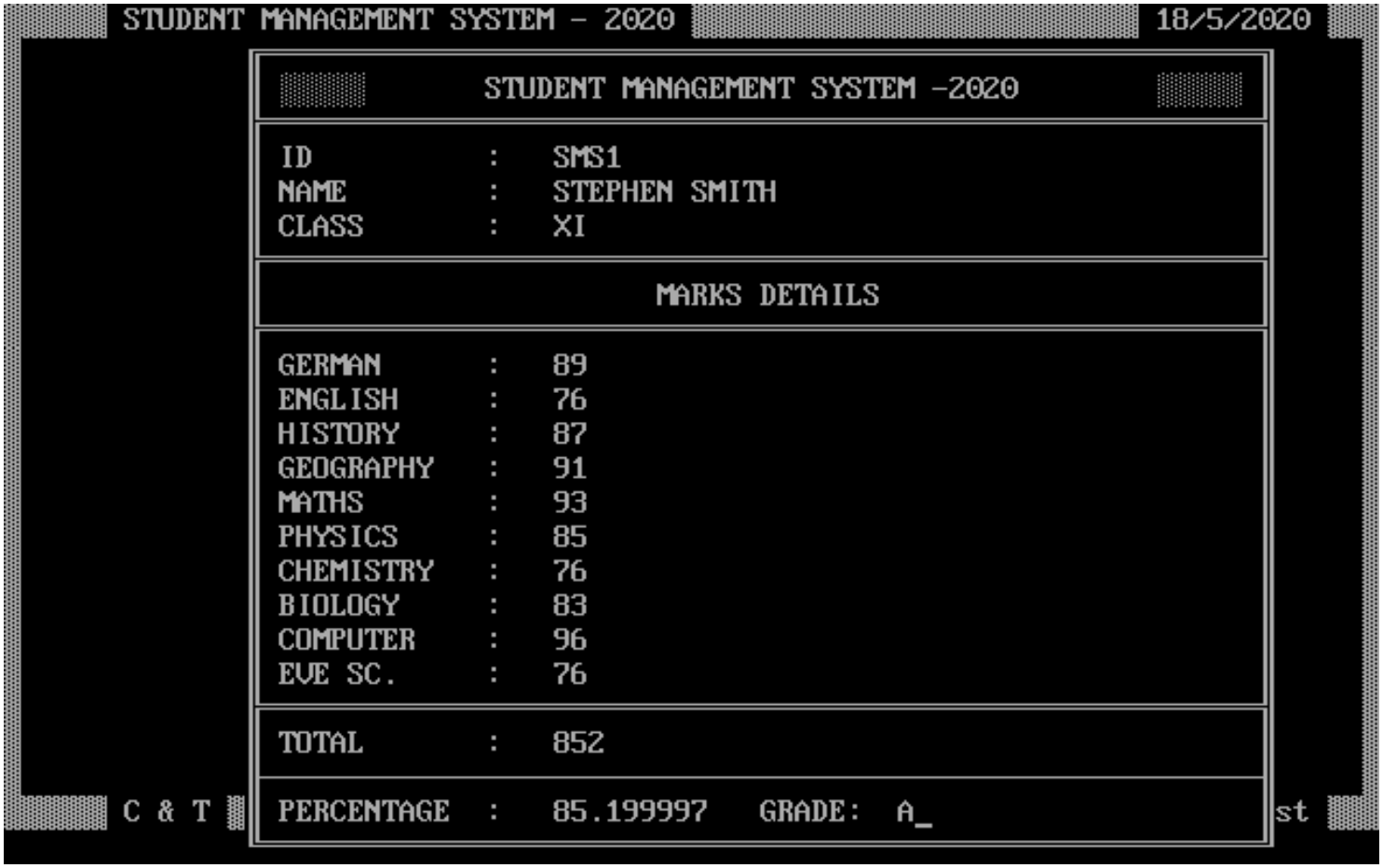
Marks Entry Form (Screenshot-11)



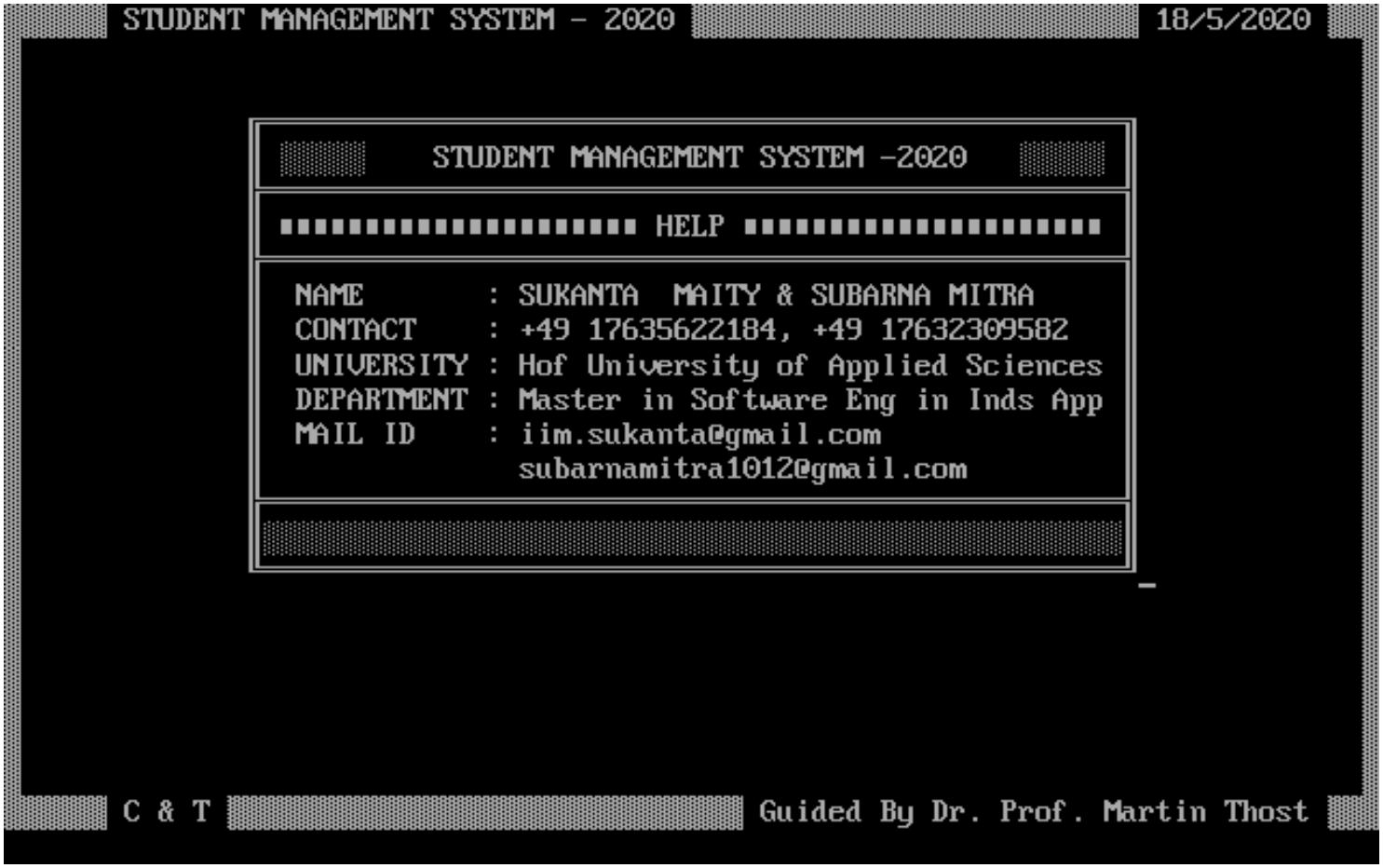
Fees Collections (Screenshot-12)



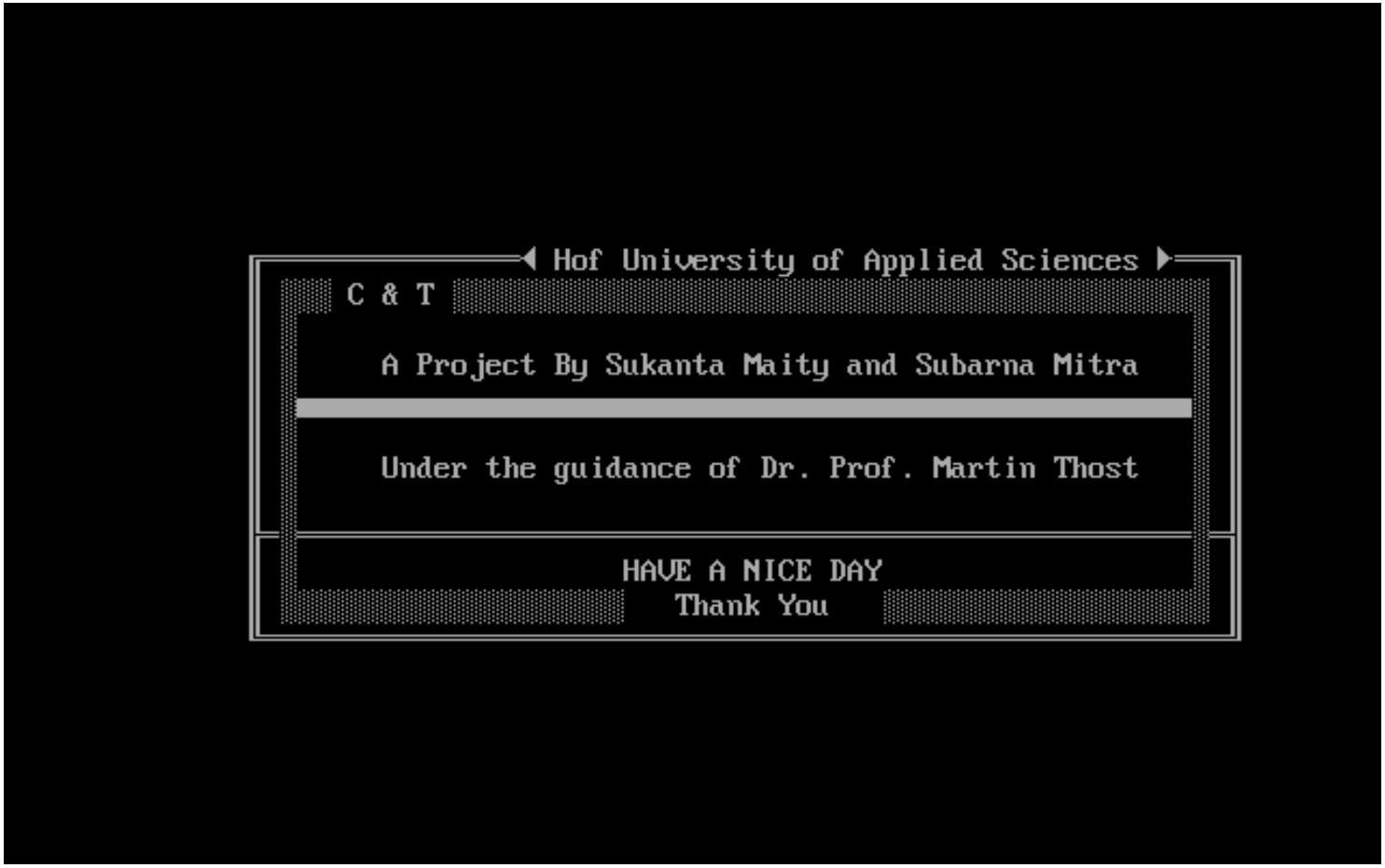
Marks Details (Screenshot-13)



Help Window (Screenshot-14)



Exit Window (Screenshot-15)

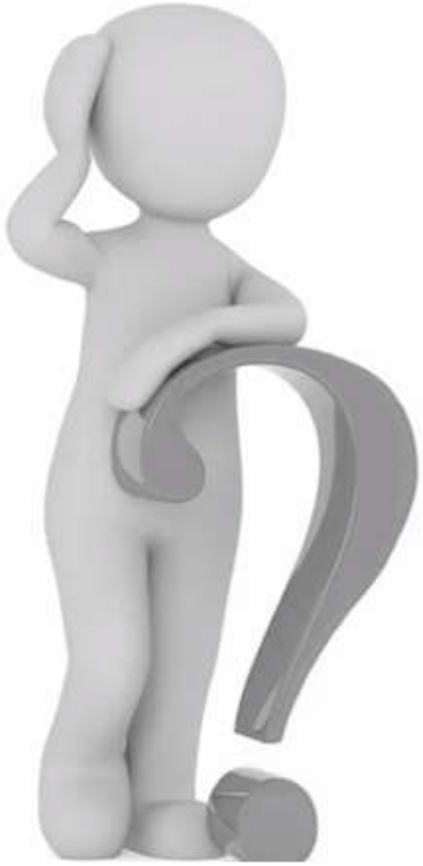


Summary

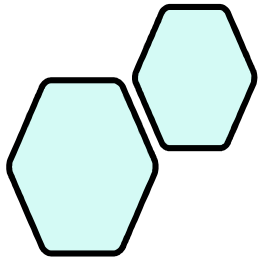
- Successful on a large scale
- Success must be sustained
- Focus on significant changes
- Usage of the recent features
- Follow Core guideline

References

- https://medium.com/@cancerian0684/what-are-four-basic-principles-of-object-oriented-programming-645af8b43727
- https://www.chu.cam.ac.uk/people/view/bjarne-stroustrup/
- https://www.modernescpp.com/index.php/c-20-an-overview
- Source of image:https://cppcodetips.wordpress.com/2019/08/31/blocking-que-implementation-in-c/
- https://cppcodetips.wordpress.com/2019/08/31/blocking-que-implementation-in-c/
- https://www.google.com/search?q=logo+of+codeblock&sxsrf=ALeKk03uEzd_CVFubYqkJrs-hgTPSNu2EA:1590965381831&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjb-Lvzl9_pAhVKiYsKHd_hC_YQ_AUoAXoECBAQAw&biw=1536&bih=674#imgrc=5mUcmww1WYKuSM
- http://www.codeblocks.org/downloads/binaries
- http://www.salaryexplorer.com/



hochschule
hof
University of Applied Sciences



Thank You