Lecture 3: Variables, Data Types & Input

Class	Programming Fundamentals
Type	Lecture
Materials	https://ecomputernotes.com/cpp/introduction-to-oop/keywords-and-identifiers https://data-flair.training/blogs/tokens-in-cpp/.htm https://medium.com/@hannahpsmith1/highlo-languages-d389ff21b4b2
Reviewed	☐
Date	@October 10, 2023

🚨 Variables**:** The containers for storing data values are known as variables. They are assigned memory randomly. They hold one value at a time.

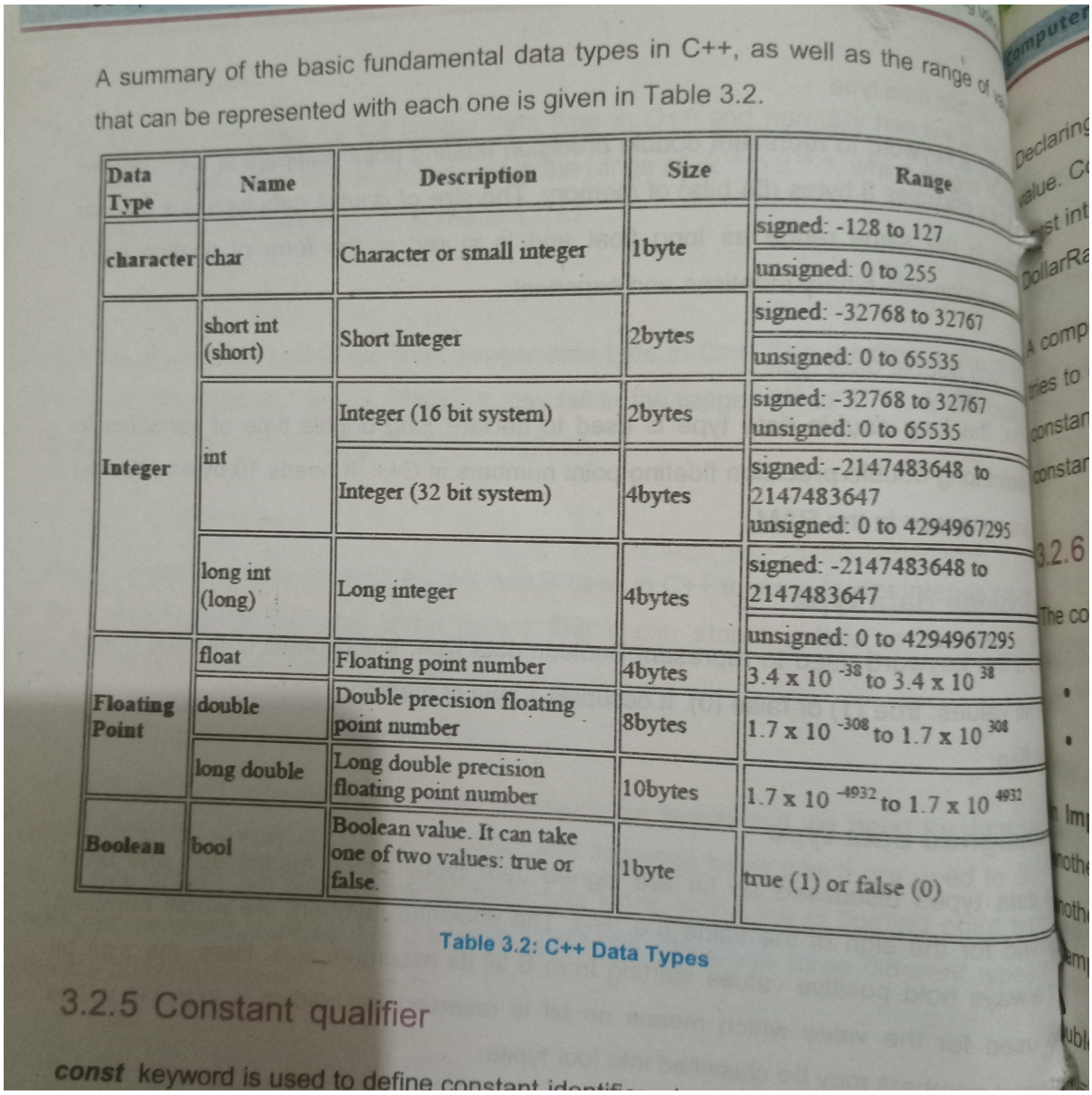
▼ Rules for Naming Variables

There are some rules for naming variables, also known as naming conventions.

1. A valid name is a sequence of one or more letters, digits, or underscores (_).
2. Neither spaces nor punctuation marks or symbols can be a part of a variable name.
3. Variable names must always start with a letter.
4. In no case, a variable can begin with a digit.
5. Names should be meaningful.
6. Names should not be too long
7. A reserved word of the C++ language cannot be used as a variable name. Here is a list of C++ keywords shown in the image below.

alignas	alignof	asm	auto	bool	break
case	catch	char	char16_t	char32_t	class
const	constexpr	const_cast	continue	decltype	default
delete	double	do	dynamic_cast	else	enum
explicit	export	extern	FALSE	float	for
friend	goto	if	inline	int	long
mutable	namespace	new	noexcept	nullptr	operator
private	protected	public	register	reinterpret_cast	return
short	signed	sizeof	static	static_assert	static_cast
struct	switch	template	this	thread_local	throw
TRUE	try	typedef	typeid	typename	union
unsigned	using	virtual	void	volatile	wchar_t
while	–	–	–	–	–

▼ Data Types



A summary of the basic fundamental data types in C++, as well as the range of va
that can be represented with each one is given in Table 3.2.

Data Type	Name	Description	Size	Range
character	char	Character or small integer	1byte	signed: -128 to 127 unsigned: 0 to 255
Integer	short int (short)	Short Integer	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
	int	Integer (16 bit system)	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
		Integer (32 bit system)	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
	long int (long)	Long integer	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
Floating Point	float	Floating point number	4bytes	3.4×10^{-38} to 3.4×10^{38}
	double	Double precision floating point number	8bytes	1.7×10^{-308} to 1.7×10^{308}
	long double	Long double precision floating point number	10bytes	1.7×10^{-4932} to 1.7×10^{4932}
Boolean	bool	Boolean value. It can take one of two values: true or false.	1byte	true (1) or false (0)

Table 3.2: C++ Data Types

3.2.5 Constant qualifier

const keyword is used to define constant identifi

Following is some code to demonstrate the different data types.

```
#include<iostream>
using namespace std;
int main()
{
  //Declaration an integer variable
```

```
  int x;
  // Initializing the above variable
  x = 19
  // Declaring and initializing variable at same time
  int y = 0;
  // Declaring a char variable
  char intial = 'A';
  // Declaring a float variables
  float dec = 2.4;
  return 0;
}
```

▼ Input

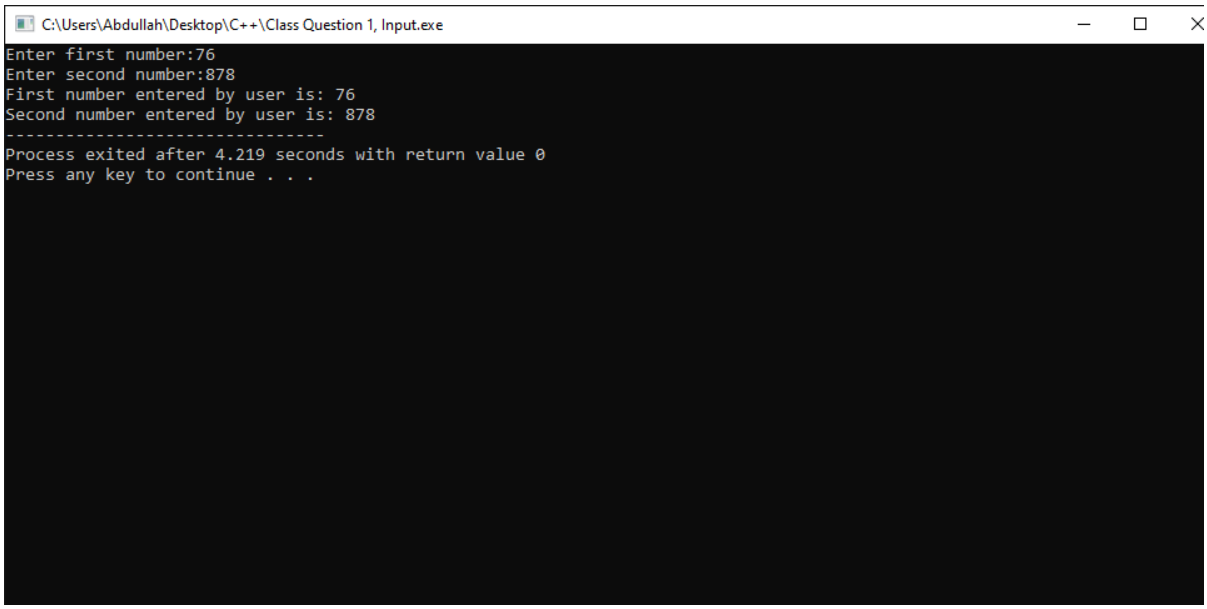
The standard input device for entering data is keyboard. In C++, input statements use cin with the extraction operator “>>”.

CQ: Write a program that takes two int variables. Take input from the user and display values on the console.

Code Ans:

```
#include<iostream>
using namespace std;
int main()
{
  //Declaring and initializing variables
  int marks1 = 0;
  int marks2 = 0;
  cout<<"Enter first number:";
  cin>>marks1;
  cout<<"\nEnter second number:";
  cin>>marks2;
  //Displaying both numbers on screen
  cout<<"First number entered by user is: "<<marks1<<endl;
  cout<<"Second number entered by user is: "<<marks2;
  return 0;
}
```

Output:



▼ Research Tasks

▼ setw()

The setw() manipulator causes the number or string that follows it to be printed within a field of x characters which is set in setw manipulator. Header file for this manipulator is <iomanip.h>.

Example:

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
  //Declaring and initializing variables
  int x=2,y=9,z=6;
  cout<<setw(8)<<"Number"<<setw(20)<<"Square"<<endl;
  cout<<setw(8)<<"2"<<setw(20)<<x*x<<endl;
  cout<<setw(8)<<"9"<<setw(20)<<y*y<<endl;
  cout<<setw(8)<<"6"<<setw(20)<<z*z<<endl;
  return 0;
}
```

▼ High Level vs Low Level

S.NO	HIGH LEVEL LANGUAGE	LOW LEVEL LANGUAGE
1.	It is programmer friendly language.	It is a machine friendly language.
2.	High level language is less memory efficient.	Low level language is high memory efficient.
3.	It is easy to understand.	It is tough to understand.
4.	It is simple to debug.	It is complex to debug comparatively.
5.	It is simple to maintain.	It is complex to maintain comparatively.
6.	It is portable.	It is non-portable.
7.	It can run on any platform.	It is machine-dependent.
8.	It needs compiler or interpreter for translation.	It needs assembler for translation.
9.	It is used widely for programming.	It is not commonly used now-a-days in programming.