Programming in C++

Computer Model

Curtis Larsen

Utah Tech University—Computing

Spring 2025

Objectives

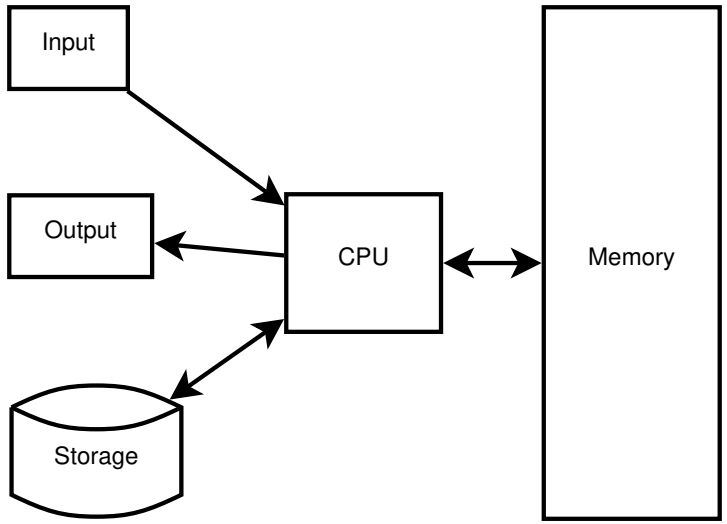
Objectives:

- Understand C++ Computer Model
- Understand C++ Memory Model
- Create Programs Utilizing C++ Computer Model
- Create Programs Utilizing C++ Memory Model

Computer Model

Computer Model

Element	Purpose	Example
Input	Receive information	keyboard,mouse,network
Output	Send information	monitor,printer,network
CPU	Execute commands	AMD Ryzen 7 9700X, M4
Memory	Temporary data storage	RAM
Storage	Persistent data storage	SDD



CPU Usage

```
int main() {
  int x = 1;
  int y = 2;
  int z;
  z = x + y; // <-- CPU instruction
  return z;
}
```

Input/Output

```
#include <iostream>
#include <string>

int read_user_number(const std::string& prompt) {
  int value;
  // output
  std::cout << prompt;
  // input
  std::cin >> value;
  return value;
}
```

Storage

```
#include <fstream>
#include <string>

void write_user_file(const std::string& file_name,
                     const int number) {
  // storage
  std::ofstream output_file(file_name);
  if(output_file) {
    output_file << "Here's your sum: ";
    output_file << number << std::endl;
    output_file.close();
  }
}
```

Memory

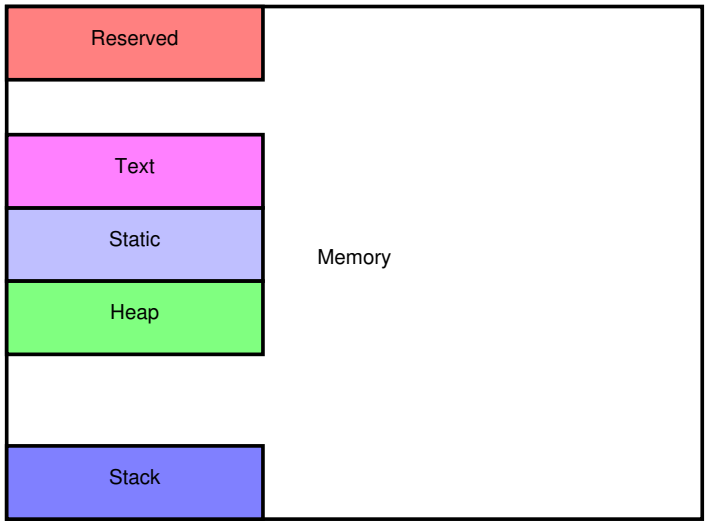
```
int main() {
  // memory
  int x = read_user_number("X? ");
  int y = read_user_number("Y? ");
  int sum = x + y;
  std::string file_name = read_user_string("File? ");

  write_user_file(file_name, sum);
  return 0;
}
```

Memory Model

Memory Model

Element	Purpose	Example
Reserved	Unusable	
Text	Code Instructions	`a + b`
Static	Persistent space	global variables
Stack	Automatic space	local function variables
Heap	Programmer managed space	expandable arrays



Text Section

```
#include <iostream>

int fred() {
  return 7;
}

int main() {
  auto fred_pointer = reinterpret_cast<void*>(&fred);

  std::cout << "Address of fred function:    "
            << fred_pointer << std::endl;

  return 0;
}
```

Static Section

```
#include <iostream>

static int g_static_variable = 10;

int main() {
  std::cout << "Address of static variable: "
            << &g_static_variable << std::endl;

  return 0;
}
```

Stack Section

```
#include <iostream>

int main() {
  int local_stack_variable = 11;

  std::cout << "Address of stack variable:  "
            << &local_stack_variable << std::endl;

  return 0;
}
```

Heap Sections

```
#include <iostream>
#include <memory>

int main() {
  std::shared_ptr<int> heap_variable =
                       std::make_shared<int>(12);
  std::cout << "Address of heap variable:   "
            << heap_variable << std::endl;

  return 0;
}
```